

Best Available Copy

Interchange.java

```
package core;
// Interchange.java - Main file for the Interchange conferencing application,
// responsible for initializations, preference setting,
// and creating the initial connection where appropriate.
//
// Copyright 1996-1998 warburg Dillon Read. All Rights Reserved.
import java.util.*;
import java.text.*;
import java.io.*;
import java.net.*;
import java.lang.Integer;
//import sun.audio.AudioPlayer;
//import sun.audio.AudioStream;

import service.startup.*;

import netscape.application.*;
import marimba.channel.ApplicationContext;
import marimba.channel.Updates;

import COM.swissbank.notification_center.*;
import COM.swissbank.util.*;
import COM.swissbank.adapter.*;
import COM.swissbank.widgets.ifc.messagebox.MessageBox;
import COM.swissbank.widgets.ifc.progress.*;

import COM.objectspace.jgl.Queue;

import util.*;
import core.User;
import service.ServiceFactory;
import service.util.ChannelRepository;
import factory.*;

import ui.controllers.channellist.*;
import preferences.*;
import service.database.*;
import ui.controllers.*;
import util.filetransfer.FileTransferController;

import adapter.service.*;
import adapter.DatabaseAdapter;
import listener.service.*;

/** Interchange conference client. Main application class which performs
 * some initializations for the application, checks for the presence of
 * the proper libraries, and starts the services of the application.
 */

public final class Interchange extends netscape.application.Application implements
ApplicationObserver,
Notifiable,
QuitListener,
WindowOwner{

    public static interface InterchangeDataProvider {
        public String getUserNick();
        public String getUsername();
        public void setUsername(String name);
        public String getUserEmail();
        public void setUserEmail(String email);
        public String getIRCServerName();
        public String getFiletransferServerName();
        public String getDatabaseServerName();
        public String getBackchatServerName();
        public String getChatwebServerName();
    }
}
```

Interchange.java

```

public String getHelpURL();
public String getBugListURL();
//Bug45 - Jiefei, redirect url
public String getRedirectURL();
// url for the whois ldap cgi
public String getWhoisURL();
public int getIRCPort();
public int getFiletransferPort();
public int getDatabasePort();
public int getBackchatPort();
public int getChatwebPort();
public Socket getIRCServerSocket() throws Exception;
public Socket getFiletransferSocket() throws Exception;
public boolean isDatabaseAvailable();
public boolean isBackChatAvailable();
public boolean isChatWebAvailable();
public boolean isRestrictedVersion();
public boolean isBrandedVersion();
public boolean isSecureVersion();
public boolean isBugListMenuShowing();
public boolean isCustomPanelAvailable();
public boolean isActiveWebLinkAvailable();
public void activeWebLink(String url);
public boolean shouldShowLicenseOnChannels();
public String getAboutTxtFileName();
}

    public static final String VersionDateString = "8/2/1999 16:30";
    public static final String Version = "2.5 Release Candidate";
    public static ThreadedNotificationCenter NC;
    public static Interchange applicationObject;
public static LicenseAgreement licenseAgreement = null ;
    private ChannelWindowFactory channelWindowFactory = null;
private Object dbShutdownLock = null;
// private Checksums checksums = null;
    private ServiceFactory serviceFactory;
private FileTransferController fileTransferController;
private PreferencesService preferencesService = null;
private BackChatService backChatService = null;
//    private Thread thread = null;
private String login = null;
private Queue qPrivates = new Queue();
private Queue qGroups = new Queue();

private static InterchangeDataProvider icDataProvider = null;
//    private Adapter adapter;
private boolean bFirstRun = false;
//JJJ private boolean bAboutContinueNotified = false;
private boolean bDoneArchiving = false;
private int dTotalSaveRequests = 0;
private int dCurrentSaveRequest = 0;

// File where the custom panel preferences are stored
public static Hashtable panelPreferences = null;

// File where the version of the Tuner the user is using is stored
public String versionFile = "version.txt";
private static String tunerVersion = null;

// File where the license agreements are found
public String agreementsFile = "agreements.txt";

    private boolean finishedInit = false;
public static NativeSound nativeSoundObject;

ICSettingsHandler icSettingsHandler = null;
ServicesDriver servicesDriver = null;

//external client interchange

```

```

Interchange.java
public Interchange(InterchangeDataProvider icdp) {
    dbShutdownLock = new Object();
    if (icdp == null) {
        throw new IllegalArgumentException("An InterchangeDataProvider must be provided");
    }
    else {
        icDataProvider = icdp;
    }
    applicationObject = this;
}

public InterchangeSettings getInterchangeSettings(){
    return icSettingsHandler.getInterchangeSettings();
}

public ChannelWindowFactory getChannelWindowFactory(){
    return this.channelWindowFactory;
}

/** Initialize the program. Reads in preference settings and launches
 * a connection if specified or preference controls to input
 * connection settings if none are provided.
 */
public void init() {
    try {
        Debug.report(this.getClass(), Debug.INFO_DEBUG, "Interchange init()");

        // Modified by raduload (10/14/98) - wanted the service factory to be started
        // early so I moved it up, and also added the ChatWebService (reload channel
        // types list.
        //if(!ICRunTimeContext.runtimeIsMarimba())
        NC = ThreadedNotificationCenter.DefaultThreadedCenter();
        Debug.report(this.getClass(), Debug.INFO_DEBUG, "Starting the Notification Center");
        NC.start();
        NC.addObserver(this, Globals.NC_COMMAND_INTERCHANGE);
        // NC.addObserver(this, Globals.USER_DEFAULT_CHANNEL_SETTINGS);

        /*
        * Instantiate a new cron object here. The cron object is in charge
of timely
time interval
        * processes. When something needs to happen at a certian time or
        * send that to the cron via the notification center. This is how
        * the channel list reload works.
        */
        new Cron(NC);

        Debug.report(this.getClass(), Debug.INFO_DEBUG, "Starting the Service Factory");

        serviceFactory = new ServiceFactory(NC);

        icSettingsHandler = new ICSettingsHandler(icDataProvider, NC);
        servicesDriver = new ServicesDriver(icDataProvider, NC,
            serviceFactory, icSettingsHandler);

        if (icDataProvider.isChatwebAvailable()) {
            serviceFactory.newService(null, service.database.ChatWebService.class);
            //BUG0230 - Jiefei, use chatweb cgi to retrieve the external client list
            serviceFactory.newService(null, service.database.ClientListService.class);
        }

        // BUG238
        // Modify by raduload on 6/24/99 - WARNING this is using chatweb service
        // info because it runs on the same machine as the chatweb cgi. We may
        // want to change that in the future.

```

```

Interchange.java
if (icDataProvider.isChatWebAvailable()) {
    serviceFactory.newService(null, service.database.MetaDirService.class);
}

if (icDataProvider.isCustomPanelAvailable()) {
    CustomPanelHandler.loadCustomPanelPreferences();
}

Debug.report(this.getClass(), Debug.INFO_DEBUG, "Registering the AboutController");
AboutController.register();

TargetFacility.sendCommand(util.Globals.COMMAND_SHOW_ABOUT_BOX_MOMENTARILY, null);

/*
 * ConnectTimer is the object that handles the display and time-delay
 * of the reconnect logic.
 */
new ConnectTimer(NC);

Debug.report(this.getClass(), Debug.INFO_DEBUG, "Registering the User Inspector
Factory");
UserInspectorFactory.register(NC);
new MessageFactory(NC);

Debug.report(this.getClass(), Debug.INFO_DEBUG, "Registering the Channel List
Controller");
channelListController.register();
Debug.report(this.getClass(), Debug.INFO_DEBUG, "Registering the Super Doc Controller");
SuperDockController.register();
Debug.report(this.getClass(), Debug.INFO_DEBUG, "Starting the Channel window Factory");
channelWindowFactory = new ChannelWindowFactory(NC);

servicesDriver.startPreferencesService();

Debug.report(this.getClass(), Debug.INFO_DEBUG, "Configuring File Transfer");
fileTransferController = FileTransferController.getFileTransferController();
NC.addObserver(fileTransferController,
util.filetransfer.FileTransferResources.COMMAND_FILE_TRANSFER_STATUS);

// Modified (added) by raduload (10/20/98) - Clients need to see our license agreement.
if (icDataProvider.shouldShowLicenseOnChannels()) {
    // new LicenseAgreement(NC, agreementsFile,
ICRunTimeContext.getMarimbaContext().getBaseURL());
    if (ICRunTimeContext.runTimeIsSpawnedJRE()) {
        licenseAgreement = new LicenseAgreement(NC, agreementsFile, new
URL(webUtilities.getStandAlonePath()));
    } else if (ICRunTimeContext.runTimeIsMarimba()) {
        licenseAgreement = new LicenseAgreement(NC,
ICRunTimeContext.getMarimbaContext().getDataDirectory() + agreementsFile, null);
    } else {
        licenseAgreement = new LicenseAgreement(NC, agreementsFile, null);
    }
}

finishedInit = true;

// start up the queued requests for private and group channels
while (!qPrivates.isEmpty()) {
    createPrivateChannel((String)qPrivates.pop());
}
while (!qGroups.isEmpty()) {
    joinChannel((String)qGroups.pop());
}

Debug.report(this.getClass(), Debug.INFO_DEBUG, "Finished Initializing Interchange");
}

```

```

                                Interchange.java
        catch(Exception e) {
    if (ICRunTimeContext.runTimeIsMarimba() && (e instanceof ClassNotFoundException)){
        Debug.report(this.getClass(), Debug.INFO_DEBUG, "class not found exception, restart
interchange");
        ICRunTimeContext.getMarimbaContext().restart();
    }
    System.out.println("Exception (init): " + e);
        e.printStackTrace();
    }
}

public void continueStartup(String _stFullName, String _stEmail) {
    Debug.report(this.getClass(), Debug.INFO_DEBUG, "Interchange continueStartup()");

    icSettingsHandler.loadInterchangeSettings(null, _stFullName, _stEmail);
    servicesDriver.startServices();
    servicesDriver.retrieveUserDefaultChannelSettings();

    if(_stFullName != null){
        bFirstRun = true;

        /*
        * The only case in which the user's full name and e-mail address is
not null is when the
        * database for this user exists. Since the e-mail is not null, no database record
        * was found. This is this user's first time using the application, save the
        * state of the application at startup.
        */
        servicesDriver.archiveApplication(channelWindowFactory);
    }

    Debug.report(this.getClass(), Debug.INFO_DEBUG, "Interchange continueStartup()
finished");
}

public String getLogin(){
    return icDataProvider.getUserNick();
}

public static InterchangeDataProvider getInterchangeDataProvider() {
    return icDataProvider;
}

public String getUserDescription() {
    if(icDataProvider != null){
        return icDataProvider.getUserName();
    }
    return getLogin();
}

public String getUserEmail() {
    if(icDataProvider != null){
        return icDataProvider.getUserEmail();
    }
    return getLogin();
}

public static String getVersionDateTimeString(){
    return VersionDateString;
}

public void saveStatus(){
    dCurrentSaveRequest++;

```

```

Interchange.java
        dTotalSaveRequests = this.channelWindowFactory.getChannelCount() + 3;
        Debug.report(this.getClass(), Debug.INFO_DEBUG, "Interchange: " + dCurrentSaveRequest + "
saves made.");
        Debug.report(this.getClass(), Debug.INFO_DEBUG, "Interchange: " + dTotalSaveRequests + "
saves needed.");

        if(dCurrentSaveRequest >= dTotalSaveRequests){
            Debug.report(this.getClass(), Debug.INFO_DEBUG, "Finished all savings");
            StatusPane.HideStatus();
            synchronized(dbShutdownLock) {
                bDoneArchiving = true;
                dbShutdownLock.notify();
            }
        }

    }

    public void reloadSettings() {
        if (servicesDriver != null)
            servicesDriver.reloadSettings();
    }

    public boolean isFinishedInit() {
        return finishedInit;
    }

    public void archiveAll(){
        bDoneArchiving = false;
        // Saving all channels, plus interchange settings, plus default channel settings, plus
checksums
        dCurrentSaveRequest = 0;
        dTotalSaveRequests = this.channelWindowFactory.getChannelCount() + 3;
        StatusPane.ShowStatus("Archiving All Open Channels...",50);
        StatusPane.getWindow().rootView().draw();
        servicesDriver.getChecksums().archive(icSettingsHandler.getInterchangeSettings(),
channelSettings.getDefaultChannelSettingsValues());
        archiveControllers();
        archiveFactories();
        servicesDriver.archiveApplication(channelWindowFactory);

        // Gotta wait for our preferences to write out so shutdown may do its thing.
        synchronized(dbShutdownLock) {
            while(!bDoneArchiving) {
                try {
                    // In case the database was not able to save some settings, timeout
                    // after 30 seconds
                    dbShutdownLock.wait(30000 + 2000*dTotalSaveRequests);
                    bDoneArchiving = true;
                } catch(Exception e) {
                    e.printStackTrace();
                }
            }
        }
        Debug.report(this.getClass(), Debug.INFO_DEBUG, "Interchange: finished ArchiveAll");
    }

    public void shutdownAll(){
        shutdownControllers();
        Debug.report(this.getClass(), Debug.INFO_DEBUG, "Interchange: Controllers shut down");
        shutdownFactories();
        Debug.report(this.getClass(), Debug.INFO_DEBUG, "Interchange: Factories shut down");
        if(service.filter.FilterEngine.getSingleton() != null)
            service.filter.FilterEngine.getSingleton().stop();
        // checksums.destroy();
        // checksums = null;
        servicesDriver.destroyChecksums();
    }

    /** This method is called to exit the Interchange application.

```

```

Interchange.java
* It is currently called from didHide on superDockController
* and in the ICMenubar code.
*/
public void exitApplication() {
    archiveAll();
    shutdownAll();
    stop();
}

private void archiveControllers() {
    TargetFacility.sendCommand(Globals.COMMAND_CHANNELLIST_ARCHIVE, null);
    TargetFacility.sendCommand(Globals.COMMAND_DOCK_ARCHIVE, null);
}

private void archiveFactories(){
    this.channelWindowFactory.archive();
}

private void shutdownControllers(){
    TargetFacility.sendCommand(Globals.COMMAND_CHANNELLIST_HIDE, null);
    TargetFacility.sendCommand(Globals.COMMAND_DOCK_SHUTDOWN, null);
}

private void shutdownFactories(){
    /*
    * Shut down the channel window factory. This will force the windows
    * to archive themselves with the autojoin set to true. This will
    * also close/kill the windows/controllers).
    */
    this.channelWindowFactory.shutdown();

    /*
    * Shut down the service factory. This will nicely shut down all of the
    * service which have been started during the applications life. By
    * nicely, the will do a join when appropriate, like the database
    * service.
    */
    this.serviceFactory.shutdown();
}

public void quit(){
    exitApplication();
}

public static void playSound(String p_stSoundName) {
    // java.net.URL soundURL;

    if (p_stSoundName != null) {
        try {
            if (ICRuntimeContext.runTimeIsSpawnedJRE() ||
                ICRuntimeContext.runTimeIsMarimba()) {
                // Code to check to see if the string is refering to a ".au" file
                // If it is make it refer to a .wav instead because of the trouble
                // we had playing sounds in Stamford. This may break cross platform
                // running of Interchange.

                String p_stOldSoundName = null;

                if (p_stSoundName.endsWith(".au")) {
                    p_stOldSoundName = p_stSoundName;
                    p_stSoundName = p_stSoundName.substring(0,
                        Page 7

```

```

Interchange.java
(p_stSoundName.length() - 2)) + "wav";
    }

    try {
        if (ICRunTimeContext.runTimeIsSpawnedJRE()) {
            nativeSoundObject.playSound(System.getProperty("marimba.channelDataDir") +
p_stSoundName);
        }
        else {
            nativeSoundObject.playSound(ICRunTimeContext.getMarimbaContext().getDataDirectory() +
p_stSoundName);
        }
    }
    catch (Exception e) {
        System.out.println("Unable to play sounds");
        // print out the error info
        e.printStackTrace();
    }
    else {
        // Runing outside of marimba
        //BUG0385 - call Main.playSound
        nativeSoundObject.playSound(p_stSoundName);
    }
    catch (Exception ie) {
        System.out.println("Cannot play sound " + p_stSoundName);
        ie.printStackTrace();
    }
}
}

```

```

public boolean performNotificationCommand(Notification _notification){
    try{
        ((Adapter)_notification.notificationObject()).apply(this);
    }
    catch(AdapterException _adapterException){
        _adapterException.printStackTrace();
    }
    return true;
}

```

```

public void run(){
    try {
        super.run();
    }
    catch (Exception e) {
        System.out.println("Exception (run): " + e);
        e.printStackTrace();
    }
    Debug.report(this.getClass(), Debug.INFO_DEBUG,"Interchange: exiting run");
}

```

```

    public void appletStopped() { super.appletStopped(); }

```

```

/*
 * Stop is defined here for marimba. We do use it when exiting the application
 * via any environment but nothing "run()" specific is done here.
 */

```


Interchange.java

```

    public void stop(){
        try {
            stopRunning();
            ThreadedNotificationCenter.DefaultThreadedCenter().stop();
            Recycler.shutdown();
            if (!finishedInit) {
                System.out.println("Interchange: didn't finish init before exiting");
            }
            if (ICRunTimeContext.runTimeIsMarimba()) { // running inside the Tuner
                System.out.println("Interchange: calling marimbaContext.stop");
                ICRunTimeContext.getMarimbaContext().stop();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void createPrivateChannel(String userNick){
        if (finishedInit) {
            PrivateChannel channel = new PrivateChannel(userNick);
            ChannelCreationController.getSingleton().channelCreationViewCreateChannel(channel);
        } else {
            qPrivates.add(userNick);
        }
    }

    public void joinChannel(String channelName) {
        if (finishedInit) {
            IRCChannel channel = new IRCChannel(channelName);
            ChannelCreationController.getSingleton().channelCreationViewCreateChannel(channel);
        } else {
            qGroups.add(channelName);
        }
    }

    // Application Observer code
    //
    // The "right" way to hook on changes in the application state is to
    // implement an ApplicationObserver which recieves callback when the
    // application will/did pause, start, or stop. In our case, we just
    // want this code in the main application logic anyway, so we just
    // make the application observe itself ... weird though that may be.

    public void applicationDidPause(netscape.application.Application app) { }
    public void applicationDidResume(netscape.application.Application app) { }
    public void applicationDidStart(netscape.application.Application app) { }
    public void currentDocumentDidChange(netscape.application.Application app, Window
win) { }
    public void focusDidChange(netscape.application.Application app, View view) { }

    /** Method to execute when the application is exiting (specifically
     *  when the application's EventLoop has finished recieving events),
     *  we want to ensure that the preferences writer threads all finish
     *  their jobs, so we'll up the priority and join them to finish.
     */
    public void applicationDidStop(netscape.application.Application app) {
        Debug.report(this.getClass(), Debug.INFO_DEBUG, "Interchange: application did stop");
    }

    // WindowOwner interface implementation
    // These should be overridden in subclasses as needed.
    public void windowDidBecomeMain(Window win) {}
    public void windowDidResignMain(Window win) {}

```

```

                                Interchange.java
    public void windowDidShow(Window win) {}
    public boolean windowWillShow(Window win) { return true; }
    public void windowWillSizeBy(Window win, Size size) {}
    public boolean windowWillHide(Window win) { return true; }
    public void windowDidHide(Window win) {}
}

```

```

// $Id: Interchange.java 2.60 1999/08/02 15:46:31 hongji new $
//
// $Log: Interchange.java $
// Revision 2.60 1999/08/02 15:46:31 hongji
// changed dates and label for 2.5 release candidate
// Revision 2.59 1999/07/26 18:24:57 hongji
// Revision 2.58 1999/07/22 21:22:29 zhounge
// Change date for beta3 release.
// Revision 2.57 1999/07/19 15:34:18 hongji
// bug fix for Bug0385
// added configuration for whoisurl
// Revision 2.53 1999/07/15 17:17:38 hongji
// Revision 2.52 1999/07/15 16:43:53 hongji
// bug0384 fix
// Revision 2.51 1999/07/09 16:38:51 hongji
// change release version to beta1
// Revision 2.50 1999/07/08 21:06:37 hongji
// added redirect url
// Revision 2.49 1999/07/08 17:49:46 hongji
// added code to retrieve external client list
// Revision 2.48 1999/07/06 21:30:52 raduload
// bug 238 - ldap integration
// Revision 2.47 1999/06/16 16:28:08 hongji
// added icdataprovder entries for text to be shown in the aboutbox
// and whether to show the license agreements on channels
// periodically
// Revision 2.46 1999/06/14 20:56:05 hongji
// added an interface activeWebLink for client interchange to
// show document
// Revision 2.45 1999/06/07 19:29:05 raduload
// new version info
// Revision 2.44 1999/06/01 22:55:29 raduload
// oops, 2 seconds not 2 milliseconds!!! hehe
// Revision 2.43 1999/06/01 21:36:04 raduload
// 30 seconds + 2 seconds per thing we need to save
// Revision 2.42 1999/05/18 14:34:06 hongji
// Revision 2.41 1999/05/12 16:10:38 hongji
// Revision 2.40 1999/05/11 14:05:18 hongji
// Revision 2.39 1999/05/10 14:03:13 hongji
// Revision 2.38 1999/05/05 17:47:35 hongji
// Revision 2.37 1999/05/03 18:22:08 zaretd
// Revision 2.36 1999/05/03 16:28:01 hongji
// Revision 2.35 1999/04/28 20:13:22 hongji
// more debug info
// Revision 2.34 1999/04/28 20:03:43 zaretd
// Revision 2.33 1999/04/27 21:57:27 hongji
// paint the statusPane for archiving IC settings
// Revision 2.32 1999/04/26 19:00:58 hongji
// Revision 2.31 1999/04/16 20:04:38 hongji
// added isBugListShowing and isCustomPanelAvailable to the
// ic data provider interface
// Revision 2.30 1999/04/16 18:34:52 hongji
// Revision 2.29 1999/04/15 20:28:16 hongji
// added bug list url to the ic data provider
// Revision 2.28 1999/04/13 19:58:34 hongji
// added getHelpURL to data provider interface
// read license agreement file from local drive instead of transmitter
// Revision 2.27 1999/04/09 14:03:23 hongji
// Revision 2.26 1999/04/06 21:06:30 hongji
// Revision 2.25 1999/03/30 16:35:17 hongji
// added chatweb support to interchangedataprovder interface.

```

```

// Interchange.java
// added code to handle ClassNotFoundException.
// Revision 2.24 1999/03/29 16:57:15 hongji
// Depending on the run time environment, loading in license
// file from different paths.
// Revision 2.23 1999/03/22 22:45:18 hongji
// createPrivateChannel() and joinChannel() will queue the requests
// if interchange initialization is not done.
// Revision 2.22 1999/03/22 19:45:32 willsos
// Rewritten to use new startup technology developed by Jiefei
// Revision 2.21 1999/02/23 21:04:23 armstrbr
// opens channelManager for new user,
// new isBackChatAvailable() call for icdataprovder
// instantiates LocalPreferenceService if !isDatabaseAvailable
// changed release date
// Revision 2.20 1999/02/19 18:38:07 raduload
// bski webutil fix
// Revision 2.19 1999/02/17 17:08:07 raduload
// a hack to fix the bug when saving preferences.
// Revision 2.18 1999/02/16 17:00:36 raduload
// timeout when exiting after 30 seconds if database was not able to
// save all the settings.
// Revision 2.17 1999/02/11 21:09:45 armstrbr
// changed license agreement to take marimba context as input parameter.
// Revision 2.16 1999/02/09 00:24:35 armstrbr
// added call to set the marimba context. i thought this was done before, but i searched
the code to no avail.
// must have been removed by accident along the way.
// Revision 2.15 1999/02/08 22:23:37 armstrbr
// added code to setUsername and setUserEmail in ic data provider because our db logic is
fucked.
// Revision 2.14 1999/02/05 22:46:09 armstrbr
// all data sources now ask icdataprovder
// Revision 2.13 1999/02/05 00:34:10 armstrbr
// uses icDataProvider to determine use for clients or not. and changed debug statements
// Revision 2.12 1999/01/20 18:52:23 raduload
// restarting when running will now work
// Revision 2.11 1999/01/19 22:53:16 raduload
// before the channel restart, it saves all your info (bug fix)
// Revision 2.10 1999/01/14 16:13:42 willsos
// Changed the CTCP Version and Date information for the final build
// Revision 2.9 1999/01/13 18:32:50 armstrbr
// removed debug statements
// Revision 2.8 1999/01/07 22:39:25 raduload
// moved down again
// Revision 2.7 1999/01/07 22:29:14 raduload
// moved channel window factory lower, because tuner couldnt start
// with it so high up. Odd
// Revision 2.6 1999/01/07 22:08:09 raduload
// fixed a timing issue where the channelwindowfactory was not being started
// soon enough.
// Revision 2.5 1999/01/07 16:35:38 raduload
// changed the date
// Revision 2.4 1999/01/06 22:23:56 armstrbr
// external client interchange web browser modifications
// Revision 2.3 1998/12/17 20:08:51 willsos
// Fixed it so that we could update Interchange's custom_panel.txt
// file without having to restart
// Revision 2.2 1998/11/19 14:53:28 raduload
// put in changes so reloading channels works
// Revision 2.1 1998/11/13 19:59:16 raduload
// added checksums
// added reloading
// fixed a continue bug, now checking for dbObject.getObject if its null to know if its a
new user
// Revision 2.0 1998/11/05 05:54:49 armstrbr
// modified InterchangeDataProviderInterface
// Revision 1.109 1998/11/04 17:10:11 raduload
// fixed a comment
// Revision 1.108 1998/11/04 17:08:57 raduload
// fixed a comment

```

```

Interchange.java
// Revision 1.107 1998/11/04 17:01:01 raduload
// put in changes, so we wait for archive to finish before starting shutdown.
// Revision 1.106 1998/11/04 14:50:15 armstrbr
// removed backchat from client interchange
// Revision 1.105 1998/11/03 22:13:09 raduload
// put in a db shutdown yield
// Revision 1.104 1998/11/03 22:09:16 raduload
// moved logic for contining from about to interchange
// Revision 1.103 1998/10/27 18:46:53 raduload
// exit code
// Revision 1.102 1998/10/27 18:16:34 raduload
// exiting code
// Revision 1.101 1998/10/27 16:47:50 raduload
// stops the running thread.
// Revision 1.100 1998/10/21 15:52:01 raduload
// changes so that client interchange people get license agreements
// every 24 hrs
// Revision 1.99 1998/10/19 20:00:16 armstrbr
// added InterchangeDataProvider interface for external interchange client user
// Revision 1.98 1998/10/15 21:55:43 raduload
// had to move startup of services to add chatwebservice
// Revision 1.97 1998/10/09 19:13:27 raduload
// nothing
// Revision 1.96 1998/10/07 21:31:59 willsos
// timezone fix
// Revision 1.95 1998/10/07 17:17:30 armstrbr
// removed references to CDT build
// modified constructor to take modes and server:ports as arguments
// Revision 1.92 1998/10/01 14:38:20 armstrbr
// changed version string
// Revision 1.91 1998/10/01 14:13:49 armstrbr
// changed version number in version string
// Revision 1.90 1998/09/24 15:36:05 raduload
// changed so channel list doesnt get loaded on start up. (set 2
// booleans to false)
// Revision 1.88 1998/09/18 18:51:06 raduload
// fixed the location of the custom panel pref file (only affects developers)
// changed some out dated text info.
// Revision 1.87 1998/09/11 21:27:56 armstrbr
// made changes to copy xml files over.
// Revision 1.85 1998/08/24 21:31:22 willsos
// custom panel code
// Revision 1.84 1998/08/20 15:17:36 armstrbr
// changed version string
// Revision 1.83 1998/06/08 21:27:12 willsos
// fixes
// Revision 1.82 1998/06/03 20:18:30 willsos
// change DEBUG and local dd deletion code
// Revision 1.81 1998/05/27 22:40:49 armstrbr
// added flags for external client desktop version. EXTERNAL_CDT
// Revision 1.80 1998/05/21 15:14:14 willsos
// CDT and Client Interchange changes
// Revision 1.79 1998/05/08 14:36:01 borad
// Use the incoming command line args to start up multiple back ends
// Revision 1.78 1998/05/04 16:02:13 borad
// Changed the object type from APP to IRC when trying
// to retrieve the default channel settings from the database
// Revision 1.77 1998/04/27 20:02:43 willsos
// made changes
// Revision 1.76 1998/04/24 01:30:21 willsos
// VERSION!!!!!!!!!!!!!!!!!!!!!!
// Revision 1.75 1998/04/23 14:38:02 willsos
// made sound changes
// Revision 1.74 1998/04/22 21:04:16 willsos
// fix to get it working for standalone
// Revision 1.73 1998/04/21 17:56:08 armstrbr
// release number
// Revision 1.72 1998/04/20 23:10:40 willsos
// DEBUG
// Revision 1.71 1998/04/17 20:45:32 willsos

```

Interchange.java

```
// version
// Revision 1.70 1998/04/15 20:50:38 willsos
// release number
// Revision 1.69 1998/04/15 18:32:45 borad
// Assign the file to null after deleting it. I think file handles were
// being kept around.
// Revision 1.68 1998/04/15 16:00:04 borad
// Commented out the code which preserves the channel list for privates
// and filters between restarts.
// Revision 1.67 1998/04/15 15:14:57 willsos
// Revision 1.66 1998/04/14 01:24:30 borad
// release 7
// Revision 1.65 1998/04/09 13:25:57 willsos
// changed clone methods and constructors
// Revision 1.64 1998/04/07 20:28:06 borad
// Shutdown the recycler
// Revision 1.63 1998/04/07 19:42:27 borad
// Remove the local prefs on startup and on restart. The startup
// portion must be removed before release into production
// Revision 1.62 1998/04/07 14:38:29 borad
// Changed the version number
// Revision 1.61 1998/04/06 21:51:06 borad
// Changed the code to point to the production database
// Revision 1.60 1998/04/01 22:31:25 borad
// Added a target-chain event to shutdown the dock
// Revision 1.59 1998/04/01 19:07:55 borad
// Added the appliction extension to the dbobject
// Revision 1.58 1998/03/31 23:24:35 borad
// Added status pane code
// Revision 1.57 1998/03/31 16:47:51 borad
// Changed the release number to beta release 4
// Revision 1.56 1998/03/27 00:54:12 borad
// Changed the version number
// Revision 1.55 1998/03/25 21:22:33 borad
// Point to the production backchat database
// Revision 1.54 1998/03/25 12:50:09 armstrbr
// changed version
// Revision 1.53 1998/03/25 18:08:27 borad
// Fully qualified server for preferences information
// Revision 1.52 1998/03/23 17:22:34 borad
// version number...
// Revision 1.51 1998/03/20 21:28:53 borad
// Save the preferences in the marimba restart code.
// Set the default hashtable for startup code
// Revision 1.50 1998/03/19 23:37:29 borad
// Added debugging code for saving the state of the applicaiton
// Revision 1.49 1998/03/18 12:37:04 willsos
// sound fixes
// Revision 1.48 1998/03/13 12:27:52 willsos
// fix the exception handler
// Revision 1.47 1998/03/12 20:43:10 borad
// Note when the app is starting for the first time
// Revision 1.46 1998/03/11 21:25:33 borad
// Changed a hashmap into a hashtable
// Revision 1.45 1998/03/11 16:39:33 borad
// Fixed the null bug
// Revision 1.44 1998/03/11 00:42:30 borad
// Added code to automatically save the applicaiton the first
// time the user logs in
// Revision 1.43 1998/03/09 21:32:55 borad
// Removed ldap stuff
// Revision 1.42 1998/03/09 15:01:22 willsos
// added plugin code
// Revision 1.41 1998/03/08 18:03:01 borad
// Setup default channel settings.
// Revision 1.40 1998/02/27 21:19:19 borad
// Changed the interface to the channel repository to be member based
// Revision 1.39 1998/02/25 09:47:57 willsos
// Moved the update code into the init method and it appears to work.
// Also did a little cleanup
```

```

// Interchange.java
// Revision 1.38 1998/02/20 22:54:31 armstrbr
// added save settings feature
// Revision 1.37 1998/02/12 21:06:43 borad
// Don't register the channelCreation Controller
// Revision 1.36 1998/02/10 22:44:27 borad
// Changed the login controller location
// Revision 1.35 1998/02/10 21:05:19 borad
// Modified the position of the startPreferences statement to be after
// the controllers are created
// Revision 1.34 1998/02/09 23:57:56 willsos
// Fixed images
// Revision 1.33 1998/02/09 23:15:02 willsos
// Updated image stuff and removed the autoupdate code
// Revision 1.31 1998/02/09 16:53:29 willsos
// Fixed the thread running issues
// Revision 1.30 1998/02/05 22:34:59 borad
// Changes for archiving private and filtered channels created.
// For the channel list
// Revision 1.29 1998/02/03 21:33:10 willsos
// Took out preferences stuff
// Revision 1.28 1998/01/27 15:29:42 borad
// Register the channel creation controller
// Revision 1.27 1998/01/23 17:15:34 borad
// Added an abstract method to selectively clean objects rather than
// just clear them
// Revision 1.26 1998/01/21 22:42:26 borad
// Change shutdown order.
// Revision 1.25 1998/01/13 22:16:38 borad
// Fixed date string to use the calendar. Also, moved the generation
// of the date out of the about view and into Interchange.java
// Revision 1.24 1998/01/12 21:17:43 borad
// Revision 1.23 1998/01/09 18:01:21 borad
// Revision 1.22 1998/01/08 21:55:18 borad
// Changed database and backchat to services.
// Revision 1.21 1998/01/06 22:14:45 borad
// Revision 1.20 1998/01/05 21:31:38 gullito
// added FileTransferController
// Revision 1.18 1998/01/02 15:34:29 borad
// apdater.apply() now throws an exception
// Revision 1.17 1997/12/22 21:01:55 borad
// Integrated back chat
// Revision 1.16 1997/12/18 20:43:43 borad
// Changed for the new database interface
// Revision 1.15 1997/12/17 22:45:47 borad
// removed refs to prefs
// Revision 1.14 1997/12/17 18:46:56 borad
// Made interchange work with new database connection
// Revision 1.13 1997/12/16 13:51:22 borad
// Added dock
// Revision 1.12 1997/12/11 16:56:51 zaretd
// Revision 1.11 1997/12/09 20:15:25 borad
// Changes for the marimba compatibility
// Revision 1.10 1997/12/04 21:31:19 lish
// Revision 1.9 1997/12/04 18:07:20 lish
// Revision 1.8 1997/12/02 19:09:41 borad
// Added the channel list controller
// Revision 1.7 1997/12/01 22:25:17 borad
// Pass in the jdbc host name to the jdbcPreferencesReader
// Revision 1.6 1997/12/01 19:10:43 borad
// Revision 1.5 1997/11/26 17:11:48 borad
// Fixes for standalone
// Revision 1.4 1997/11/20 22:34:28 borad
// not sure
// Revision 1.3 1997/11/18 22:39:22 borad
// Modification for compiling 2.0
// Revision 1.2 1997/11/06 20:42:37 borad
// added package information
// Revision 1.1 1997/11/06 18:12:32 borad
// Initial revision
// Revision 2.10 1997/11/06 17:29:43 willsos

```

Interchange.java

```
// Changed to use the new class paths
// Revision 2.9 1997/11/03 21:47:19 borad
// Revision 1.46 1997/11/03 19:05:28 borad
// Really changed the version to 1.3.4
// Revision 1.45 1997/11/03 19:02:33 borad
// Changed the version to 1.3.4
// Revision 1.44 1997/10/30 22:47:06 willsos
// Revision 1.43 1997/10/30 15:20:42 zaretd
// Revision 1.42 1997/10/29 17:11:21 zaretd
// Revision 1.41 1997/10/29 16:26:16 borad
// Changed startup code to accomodate login panel and changed
// the wording on the update panel to be more readable
// Revision 1.40 1997/10/28 16:03:50 borad
// Upped the version number
// Revision 1.39 1997/10/22 17:37:45 willsos
// Revision 1.38 1997/10/21 20:05:25 willsos
// Revision 1.37 1997/10/16 18:43:55 willsos
// Revision 1.36 1997/10/16 18:38:01 willsos
// Revision 1.35 1997/10/16 15:20:52 gullieto
// Revision 1.34 1997/10/08 18:01:52 willsos
// Added the fontNamed method to fix bug in IFC
// Revision 1.33 1997/10/08 14:56:12 borad
// Fixed the start bug
// Revision 1.32 1997/10/07 14:22:57 willsos
// Commented out the sleep code to insert the PWF in full time
// Revision 1.31 1997/10/03 19:35:35 borad
// shutdown the channelwindowfactory
// Revision 1.30 1997/10/03 17:52:42 borad
// Recieve notification if our user had to have a nick change
// applied (borad2)
// Revision 1.29 1997/10/02 19:14:40 willsos
// Revision 1.28 1997/10/02 18:45:35 willsos
// Added the PreferencesWriterController
// Revision 1.27 1997/10/02 18:01:09 borad
// Added method to gain access to the channelwindowfactory
// Revision 1.26 1997/10/01 21:37:12 borad
// Added an arg for the back end
// Revision 1.25 1997/09/30 15:28:41 borad
// Chaneg irc notification string. Change the way the
// service factory is started.
// Revision 1.24 1997/09/26 16:31:35 zaretd
// font fix
// Revision 1.23 1997/09/26 08:50:06 willsos
// Made additions to sound so that it will at least work on JRE.
// Still getting error code in Marimba
// Revision 1.22 1997/09/24 12:45:56 willsos
// NO Changes
// Revision 1.21 1997/09/23 20:20:22 borad
// start stuff
// Revision 1.20 1997/09/23 17:52:12 borad
// more shutdown testing
// Revision 1.19 1997/09/22 21:10:35 borad
// Put in shutdown code for marimba and jre
// Revision 1.18 1997/09/19 14:35:54 borad
// Added the instantiation of ConnectTimer to the init() method
// Revision 1.17 1997/09/19 14:01:49 zaretd
// Revision 1.16 1997/09/18 14:47:21 gullieto
// removed the other thread - still don't know why it was ever there!
// Revision 1.15 1997/09/15 21:27:27 borad
// Changed the thread startup code to not -2 the main thread
// priority. Added notification to the service when stopping the
// application through marimba
// Revision 1.14 1997/09/12 19:36:05 borad
// Modified exit code and changed version number
// Revision 1.13 1997/09/10 15:28:34 borad
// Added the autojoin channels list to the hashtable arg list passed
// into the service factory
// Revision 1.12 1997/09/09 20:19:45 borad
// Added an argument to the hash of args passed into the instantiation
// of a new service
```

```

// Interchange.java
// Revision 1.11 1997/09/05 15:39:00 borad
// New Back End integration
// Revision 1.10 1997/08/26 20:02:44 mjt
// created isTuner boolean variable to simplify code conditionals inside
// or not of the Tuner environment.
// Revision 1.9 1997/08/22 15:47:50 mjt
// various fixes, additions for running in marimba context
// Revision 1.8 1997/08/13 16:45:59 borad
// more changes for marimba
// Revision 1.7 1997/08/13 16:30:29 borad
// Fixed sounds for marimba
// Revision 1.6 1997/08/13 15:37:57 mjt
// added some supporting methods for Castanet
// Revision 1.5 1997/08/12 19:38:27 mjt
// changed version string to include invite_only
// Revision 1.4 1997/08/05 23:55:02 mjt
// fixed the Cafe case of the sound to be able to find sound using
// the default IFC Sound.soundNamed() method
// Revision 1.3 1997/08/05 21:41:59 borad
// Revision 1.2 1997/08/01 18:45:44 mjt
// changed IFC version to report 1.1 instead of 1.1.1 to
// get it to work as a FLOAT
// Revision 1.1 1997/07/24 19:48:12 ishuevm
// Initial revision
// Revision 1.49 1997/05/14 19:23:18 zaretd
// Changed version to 1.1.
// Revision 1.48 1997/05/12 19:26:27 seymour
// Removed IFC version checking debugs, upped version to 1.1b5
// Revision 1.47 1997/05/12 18:54:54 seymour
// Added additional check on IFC version parsing.
// Revision 1.46 1997/05/12 18:34:17 seymour
// Removed application observer code.
// Revision 1.45 1997/05/12 18:06:35 seymour
// Added check on IFC version, turned off debugging, and made
// version date creation more efficient.
// Revision 1.44 1997/05/08 22:20:45 seymour
// Removed debugging for focusDidChange, which was spamming.
// Revision 1.43 1997/05/08 17:28:33 seymour
// Added debugging for application observer methods, removed
// deprecated methods (getValue and processEvent), turned
// debuggin on and upped version to 1.1a5.
// Revision 1.42 1997/05/07 20:28:54 borad
// Bumped up the version number b4
// Revision 1.41 1997/05/05 19:47:50 borad
// version 1.1b3
// Fix to kick
// Revision 1.40 1997/05/01 18:25:53 seymour
// Removed thread debugging on appDidStop.
// Revision 1.39 1997/05/01 14:31:49 seymour
// Added thread scanning code on appDidStop to show state of threads
// at shutdown, removed debugging prints for version date.
// Revision 1.38 1997/04/30 16:41:50 seymour
// Fixed bug in date parsing.
// Revision 1.37 1997/04/30 16:03:53 seymour
// Disabled PWF, set version to 1.1b2
// Revision 1.36 1997/04/29 21:40:43 seymour
// Enabled PWF again.
// Revision 1.35 1997/04/29 20:46:15 seymour
// Disabled PWF.
// Revision 1.34 1997/04/29 20:35:18 seymour
// Updated version number to 1.1a2.
// Revision 1.33 1997/04/29 18:40:04 borad
// Changed NC_COMMAND_EXIT_APPLICATION to
// NC_COMMAND_WILL_EXIT_APPLICATION
// Revision 1.32 1997/04/29 15:42:55 seymour
// Removed call to FinishWriting on application stop, this function moved
// to PreferencesWriterFinisher object.
// Revision 1.31 1997/04/25 02:12:26 seymour
// Made application observer of self and added method to wait for
// preferences writer threads to finish before completing execution.

```


Interchange.java

```
// Also removed opening of www page.  
// Revision 1.30 1997/04/19 02:17:15 seymour  
// Updated version to 1.01b, all functionality is though to be in now,  
// though some bugs and untested code remain.  
// Revision 1.29 1997/04/12 21:11:22 seymour  
// Cleaned up initialization methods and made RCS comments not  
// show file pathnames.
```

```

                                ChannelDisplaySettingsView.java
package ui.views.displaysettings;

```

```

import netscape.application.*;
import COM.swissbank.widgets.ifc.core.CustomView;
import COM.swissbank.widgets.ifc.misc.*;
import COM.swissbank.widgets.ifc.buttons.*;
import COM.swissbank.widgets.ifc.floatinghints.*;
import COM.swissbank.util.ifc.WindowManager;
import preferences.*;
import ui.views.channel.*;
import core.*;

```

```

/**
 *      This is the view class for the above dialog.
 */
public class ChannelDisplaySettingsView extends CustomView implements FloatingHintConsumer {
// BUG0250 - replaced "accept all" and "accept changes" with a single ACCEPT_COMMAND
    public static final String      ACCEPT_ALL_COMMAND = "accept";
// public static final String      ACCEPT_CHANGES_COMMAND = "accept changes";
    public static final String      CANCEL_COMMAND = "cancel";
    public static final String      LINK_COLOR = "link_color";
    public static final String      BACKGROUND_COLOR = "background_color";

    private ContainerView
    private ContainerView
    private ContainerView
    containerViewLinkColorModifier;
    private ContainerView
    private ContainerView
    private ScrollGroup
    messageDisplayScrollGroup;

                                containerViewBGColor;
                                containerViewLinkColor;
                                containerViewBGColorModifier;

                                messageDisplayContainer;
                                settingsContainer;

    private FloatingHintColorWell
    private FloatingHintColorWell
                                backgroundColorWell;
                                linkColorWell;

    private DisplayPropertiesView
    private DisplayPropertiesView
    private DisplayPropertiesView
                                selfProps;
                                otherProps;
                                alertProps;

    private ChannelSettings
    private InterchangeSettings
    Application.application().getInterchangeSettings();
                                channelSettings;
                                interchangeSettings = ((Interchange)

//    private ExtendedButton
//    private ExtendedButton
//    private ExtendedButton
                                acceptAllBtn;
                                acceptChangesBtn;
                                cancelBtn;

    private String
    stChannelName;
    private MessageDisplayTextView messageDisplayTextView;

    protected FloatingHintConsumer curConsumer = null;
    protected FloatingHintProvider floatingHintProvider;

    public void setFloatingHintProvider(FloatingHintProvider prov) { floatingHintProvider
= prov; }
    public FloatingHintProvider getFloatingHintProvider() { return floatingHintProvider;
}

    public ChannelDisplaySettingsView(String _stChannelName){
        super();
        this.stChannelName = _stChannelName;
        resetSizes();
    }

```

ChannelDisplaySettingsView.java

```
/**
 *      From the FloatingHintProvider interface.
 */
public void startHint(FloatingHintConsumer consumer, String context) {
    curConsumer = consumer;
    if (floatingHintProvider != null)
        floatingHintProvider.startHint(this, null);
}

/**
 *      From the FloatingHintProvider interface.
 */
public void endHint(FloatingHintConsumer consumer) {
    curConsumer = null;
    if (floatingHintProvider != null)
        floatingHintProvider.endHint(this);
}

/**
 *      From the FloatingHintConsumer interface.
 */
public void customizeFloatingHint(FloatingHint floatingHint) {
    if (curConsumer != null)
        curConsumer.customizeFloatingHint(floatingHint);
}

/**
 *      From the FloatingHintConsumer interface - unused.
 */
public void setHintString(String str) {}
public String getHintString() {return null;}

public void createWidgets() {
    this.containerViewBGColor = new ContainerView();
    this.containerViewBGColor.setTitle("Background Color");
    addSubview(this.containerViewBGColor);

    this.containerViewBGColorModifier = new ContainerView();
    this.containerViewBGColor.addSubview(containerViewBGColorModifier);

    this.containerViewLinkColor = new ContainerView();
    this.containerViewLinkColor.setTitle("Html Link/Channel Link Color");
    addSubview(this.containerViewLinkColor);

    this.containerViewLinkColorModifier = new ContainerView();
    this.containerViewLinkColor.addSubview(this.containerViewLinkColorModifier);

    this.messageDisplayTextView = new MessageDisplayTextView();
    this.messageDisplayTextView.setEditable(false);
    this.messageDisplayTextView.setSelectable(true);
    this.messageDisplayTextView.moveTo(0,0);

    this.messageDisplayScrollGroup = new ScrollGroup();
    this.messageDisplayScrollGroup.setVertScrollBarDisplay(ScrollGroup.AS_NEEDED_DISPLAY);
    this.messageDisplayScrollGroup.setBorder(BezelBorder.loweredBezel());
    this.messageDisplayScrollGroup.setBackgroundColor(Color.white);
    //this.messageDisplayScrollGroup.setBuffered(true);
    this.messageDisplayScrollGroup.setContentView(this.messageDisplayTextView);
    addSubview(this.messageDisplayScrollGroup);

    settingsContainer = new ContainerView();
    Page 2
}
```

```

        ChannelDisplaySettingsView.java
        settingsContainer.setTitle("Event Settings");
        addSubview(settingsContainer);

        backgroundColorWell = new FloatingHintColorWell();
        backgroundColorWell.setFloatingHintProvider(this);
        backgroundColorWell.setTarget(this);
        backgroundColorWell.setCommand(BACKGROUND_COLOR);
        backgroundColorWell.setHintString("Click to launch the color chooser");
        containerViewBGColorModifier.addSubview(backgroundColorWell);

        linkColorWell = new FloatingHintColorWell();
        linkColorWell.setFloatingHintProvider(this);
        linkColorWell.setTarget(this);
        linkColorWell.setCommand(LINK_COLOR);
        linkColorWell.setHintString("Click to launch the color chooser");
        containerViewLinkColorModifier.addSubview(linkColorWell);

        selfProps = new DisplayPropertiesView("Self");
        selfProps.setFloatingHintProvider(this);
        selfProps.setChannelDisplayPropertiesView(this);
        settingsContainer.addSubview(selfProps);

        otherProps = new DisplayPropertiesView("Other");
        otherProps.setFloatingHintProvider(this);
        otherProps.setChannelDisplayPropertiesView(this);
        settingsContainer.addSubview(otherProps);

        alertProps = new DisplayPropertiesView("Alert");
        alertProps.setFloatingHintProvider(this);
        alertProps.setChannelDisplayPropertiesView(this);
        settingsContainer.addSubview(alertProps);

        acceptAllBtn = new ExtendedButton();
        acceptAllBtn.setTitle("Accept");
        acceptAllBtn.setCommand(ACCEPT_ALL_COMMAND);
        acceptAllBtn.setTarget(this);
        acceptAllBtn.setFloatingHintProvider(this);
        acceptAllBtn.setHintString("Click to accept changes");
        sizeButtonForTitle(acceptAllBtn, util.Globals.CONST_BUTTON_SIZE);
        addSubview(acceptAllBtn);

// BUG0250 - replaced "accept all" and "accept changes" with a single ACCEPT_COMMAND
/*
        acceptChangesBtn = new ExtendedButton();
        acceptChangesBtn.setTitle("Accept Changes");
        acceptChangesBtn.setCommand(ACCEPT_CHANGES_COMMAND);
        acceptChangesBtn.setTarget(this);
        acceptChangesBtn.setFloatingHintProvider(this);
        acceptChangesBtn.setHintString("Click to accept only the current changes");
        sizeButtonForTitle(acceptChangesBtn, util.Globals.CONST_BUTTON_SIZE);
        addSubview(acceptChangesBtn);
*/

        cancelBtn = new ExtendedButton();
        cancelBtn.setTitle("Cancel");
        cancelBtn.setCommand(CANCEL_COMMAND);
        cancelBtn.setTarget(this);
        cancelBtn.setFloatingHintProvider(this);
        cancelBtn.setHintString("Click to cancel");
        sizeButtonForTitle(cancelBtn, util.Globals.CONST_BUTTON_SIZE);
        addSubview(cancelBtn);
    }

    private void handleLinkColorChange(){
        this.containerViewLinkColorModifier.setBackgroundColor(Color.white);
        settingsChanged(true);
        rootView().colorChooser().hide();
    }
}

```

ChannelDisplaySettingsView.java

```
private void handleBackgroundColorChange(){
    this.containerViewBGColorModifier.setBackgroundColor(Color.white);
    settingsChanged(true);
    rootView().colorChooser().hide();
}

public void setChannelSettings(ChannelSettings _channelSettings){
    this.channelSettings = _channelSettings;
    setBackgroundColor(this.channelSettings.getBGColor());
    setLinkColor(this.channelSettings.getLinkHtmlColor());
    setSelfDisplaySettings(this.channelSettings.getSelfDisplaySettings());
    setOtherDisplaySettings(this.channelSettings.getOtherDisplaySettings());
    setAlertDisplaySettings(this.channelSettings.getAlertDisplaySettings());
    settingsChanged(false);
}

public void settingsChanged(boolean _bFromUser){
    this.channelSettings.setSelfDisplaySettings(this.selfProps.getSettings());
    this.channelSettings.setOtherDisplaySettings(this.otherProps.getSettings());
    this.channelSettings.setAlertDisplaySettings(this.alertProps.getSettings());
    this.channelSettings.setBGColor(getBackgroundColor());
    this.channelSettings.setLinkHtmlColor(getLinkColor());

    if(_bFromUser){
        this.acceptAllBtn.setLoweredColor(Color.white);
        this.acceptAllBtn.setRaisedColor(Color.white);
        this.acceptAllBtn.setDirty(true);
    }

    addSampleContent();
    resetSizes();
}

public void addSampleContent(){
    Content _content = new Content();

    this.messageDisplayTextView.clearBuffer();
    this.messageDisplayTextView.setChannelSettings(this.channelSettings);
    this.messageDisplayScrollView.setBackgroundColor(this.channelSettings.getBGColor());

    _content.setAuthorType(Content.SELF);
    _content.setChannel("#interchange");
    _content.setAuthor(this.interchangeSettings.getNickname());
    _content.setContent("The person who says it cannot be done should not interrupt the person doing it");

    this.messageDisplayTextView.appendContent(_content);

    _content = new Content();
    _content.setAuthorType(Content.OTHER);
    _content.setChannel("#interchange");
    _content.setAuthor("armstrbr");
    _content.setContent("Make three correct guesses consecutively and you will establish a reputation as an expert");

    this.messageDisplayTextView.appendContent(_content);

    _content = new Content();
    _content.setAuthorType(Content.ALERT);
    _content.setChannel("#interchange");
    _content.setAuthor("willsos");
    _content.setContent("The further backward you look, the further forward you can see.");

    this.messageDisplayTextView.appendContent(_content);
}
```

ChannelDisplaySettingsView.java

```

_content = new Content();
_content.setAuthorType(Content.OTHER);
_content.setChannel("#interchange");
_content.setAuthor("borad");
_content.setContent("Visit #interchange for Interchange related discussions or visit
http://chiweb/Interchange for more information about Interchange");

this.messageDisplayTextView.appendContent(_content);
}

public void resetSizes() {
    int workWidth = totWidth - 3*MARGIN;
    int workHeight = totHeight - 3*MARGIN;

    /*
    * Size the settingsContainer to be the height of 3 rows of buttons and the margin's
    between
    * all three rows and the top and bottom of the button rows.
    */
    settingsContainer.sizeTo(totWidth - 2* MARGIN, BUTTON_HEIGHT * 3 + MARGIN *
7);
    settingsContainer.moveTo(startPosX + MARGIN,
startPosY + totHeight - MARGIN *2 - cancelBtn.height() -
this.settingsContainer.height());

    // background color container.
    containerViewBGColor.sizeTo(this.settingsContainer.width()/2 - 2 * MARGIN,
60);
    this.containerViewBGColor.moveTo(startPosX + MARGIN,
this.settingsContainer.y() - this.containerViewBGColor.height() -MARGIN);

    this.containerViewLinkColor.sizeTo(this.containerViewBGColor.width(),
this.containerViewBGColor.height());
    this.containerViewLinkColor.moveTo(totWidth -
this.containerViewLinkColor.width() - 2 * MARGIN,
this.containerViewBGColor.y());

    workWidth = containerViewBGColor.interiorRect().width - 2*SMALL_MARGIN;
    workHeight = containerViewBGColor.interiorRect().height - 2*SMALL_MARGIN -
MARGIN;
    containerViewBGColorModifier.sizeTo(workWidth, workHeight);

    containerViewBGColorModifier.moveTo(this.containerViewBGColor.interiorRect().x +
SMALL_MARGIN,
this.containerViewBGColor.interiorRect().y);

    // now do the link color component
    workWidth = containerViewLinkColor.interiorRect().width - 2*SMALL_MARGIN;
    workHeight = containerViewLinkColor.interiorRect().height - 2*SMALL_MARGIN -
MARGIN;
    containerViewLinkColorModifier.sizeTo(workWidth, workHeight);

    containerViewLinkColorModifier.moveTo(this.containerViewLinkColor.interiorRect().x +
SMALL_MARGIN,
this.containerViewLinkColor.interiorRect().y);

    this.messageDisplayScrollGroup.sizeTo(this.settingsContainer.width(),
height() - this.containerViewBGColor.y() - 2* MARGIN);
    this.messageDisplayScrollGroup.moveTo(MARGIN, MARGIN);

```

```

ChannelDisplaySettingsView.java
cancelBtn.moveTo(startPosX + totWidth - MARGIN - cancelBtn.width(), startPosY
+ totHeight - MARGIN - cancelBtn.height());
// acceptChangesBtn.moveTo(cancelBtn.x() - MARGIN - acceptChangesBtn.width(),
cancelBtn.y());
acceptAllBtn.moveTo(cancelBtn.x() - MARGIN - acceptAllBtn.width(),
cancelBtn.y());

// now do the background color component
workWidth = containerViewBGColorModifier.interiorRect().width -
2*SMALL_MARGIN;
workHeight = containerViewBGColorModifier.interiorRect().height -
2*SMALL_MARGIN - MARGIN;
backgroundColorWell.setSize(workWidth, workHeight);
backgroundColorWell.moveTo(this.containerViewBGColorModifier.width()/2 -
backgroundColorWell.width()/2,
this.containerViewBGColorModifier.height()/2 - backgroundColorWell.height()/2);

// now do the link color component
workWidth = containerViewLinkColorModifier.interiorRect().width -
2*SMALL_MARGIN;
workHeight = containerViewLinkColorModifier.interiorRect().height -
2*SMALL_MARGIN - MARGIN;
linkColorWell.setSize(workWidth, workHeight);
linkColorWell.moveTo(this.containerViewLinkColorModifier.width()/2 -
linkColorWell.width()/2,
this.containerViewLinkColorModifier.height()/2 - linkColorWell.height()/2);

// now do the settings strips
workWidth = settingsContainer.interiorRect().width - 2*SMALL_MARGIN;
workHeight = settingsContainer.interiorRect().height;
selfProps.setSize(workWidth, BUTTON_HEIGHT);
otherProps.setSize(workWidth, BUTTON_HEIGHT);
alertProps.setSize(workWidth, BUTTON_HEIGHT);
selfProps.moveTo(settingsContainer.interiorRect().x + SMALL_MARGIN,
settingsContainer.interiorRect().y);
otherProps.moveTo(selfProps.x(), selfProps.bounds().maxY() + MARGIN);
alertProps.moveTo(otherProps.x(), otherProps.bounds().maxY() + MARGIN);

/*
 * Size and position the message display text view. This is here sample
 * test appears. Also, set the title here.
 */
workWidth = this.messageDisplayScrollGroup.width() - 2*SMALL_MARGIN;
workHeight = this.messageDisplayScrollGroup.height();
this.messageDisplayTextView.setSize(workWidth, workHeight);
//this.messageDisplayTextView.moveTo(this.messageDisplayScrollGroup.interiorRect().x +
SMALL_MARGIN,
// this.messageDisplayScrollGroup.interiorRect().y);
// this.messageDisplayScrollGroup.setTitle("Sample Text for " + this.stChannelName);
}

public void resetView(){
    this.selfProps.resetView();
    this.otherProps.resetView();
    this.alertProps.resetView();
    this.containerViewLinkColorModifier.setBackgroundColor(Color.lightGray);
    this.containerViewBGColorModifier.setBackgroundColor(Color.lightGray);
    this.acceptAllBtn.setLoweredColor(Color.lightGray);
    this.acceptAllBtn.setRaisedColor(Color.lightGray);
}

public void setupFocus() {}

public Color getBackgroundColor() {
    return backgroundColorWell.color();
}

public void setBackgroundColor(Color col) {

```

```

                                ChannelDisplaySettingsView.java
        backgroundColorWell.setColor(col);
    }

    public void setLinkColor(Color _color){
        this.linkColorWell.setColor(_color);
    }

    public Color getLinkColor(){
        return this.linkColorWell.color();
    }

    public DisplaySettings getSelfDisplaySettings() {
        return selfProps.getSettings();
    }

    public void setSelfDisplaySettings(DisplaySettings sets) {
        selfProps.setSettings(sets);
    }

    public DisplaySettings getOtherDisplaySettings() {
        return otherProps.getSettings();
    }

    public void setOtherDisplaySettings(DisplaySettings sets) {
        otherProps.setSettings(sets);
    }

    public DisplaySettings getAlertDisplaySettings() {
        return alertProps.getSettings();
    }

    public void setAlertDisplaySettings(DisplaySettings sets) {
        alertProps.setSettings(sets);
    }

    // forward to my target
    public void performCommand(String com, Object obj) {

/*
 * There has to be a better way to do this.. I am hacking
 * my way around this because I spent too long on the resize
 * code and it has to get done! -borad
 */
        if (target != null)
            target.performCommand(com, obj);

        if(obj instanceof FloatingHintColorWell){
            settingsChanged(false);
        }

        if(LINK_COLOR.equals(com)){
            handleLinkColorChange();
        }
        else if(BACKGROUND_COLOR.equals(com)){
            handleBackgroundColorChange();
        }
    }
}

```


ChannelwindowView.java

```
package ui.views.channel;
```

```
import java.awt.Toolkit;
import ui.controllers.*;
import netscape.application.*;
import netscape.util.*;

import COM.swissbank.adapter.*;
import COM.swissbank.util.*;
import COM.swissbank.notification_center.*;
import COM.swissbank.widgets.ifc.core.*;
import COM.swissbank.widgets.ifc.text.*;
import COM.swissbank.widgets.ifc.toolbar.*;
import COM.swissbank.widgets.ifc.buttons.*;
import COM.swissbank.widgets.ifc.splitter.*;
import COM.swissbank.widgets.ifc.popupmenus.*;
import COM.swissbank.util.ifc.*;
```

```
import core.*;
import ui.views.*;
import ui.views.ChannelResources;
import preferences.*;
import adapter.*;
import util.*;
import ui.controllers.channellist.ChannelCreationController;
import COM.swissbank.widgets.ifc.floatinghints.*;
import COM.swissbank.widgets.ifc.misc.LockingScrollGroup;
import COM.swissbank.widgets.ifc.misc.LockingScrollBar;
import ui.views.channel.custom.*;
```

```
/**
 * @(#)ChannelwindowView.java
 *
 * ChannelwindowView is the main channel view that all other views subclass from.
 * It contains a lot of the logic etc that the subclasses use for layout and
 * control.
 *
 * Copyright 1998 Warburg Dillon Read. All Rights Reserved.
 *
 * @author D. Zaret, Don Bora, Sean Willson
 *
 * overhauled by D. Zaret on 4/19/99
 * bugs fixed by raduload on 6/4/99
 */
public abstract class ChannelwindowView extends CustomView implements
ChannelwindowViewMenuToolbarSet.ChannelwindowViewMenuToolbarSetOwner,
```

Target,

SplitViewOwner,

TextViewOwner,
TextFieldOwner,
PopupMenuConsumer,
PopupMenuProvider,
FloatingHintProvider,

```
CustomInputPanel.CustomInputPanelOwner
{
    public static interface ChannelwindowViewOwner extends Owner {
        public void handleMakeOperator(User member);
        public void handleKick(User member);
        public void handleInvite(User user, Channel channel);
        public void handleNonSpecificPrivate();
        public void handleNonSpecificWhois();
        public void handleSpecificPrivate(User user);
    }
}
```

```

                                Channelwindowview.java
public void handlespecificwhois(User user);
    public void handleChannelPrefs();
    public void handleChannelPrefsDefault();
public void handleChannelPrefsGeneral();
    public void handleDock();
    public void handleUnDock();
    public void handleChannelClose();
    public void handleChannelClear();
    public void handlePostFile();
    public void handleGetChannel(String channelName); // for URL channel links
    public void handleShowChannelListwindow();
public void handleShowCreationView(String type);
    public void handleDockAll();
    public void handleQuitApplication();
    public void handleShowHelpContents();
    public void handleShowHelpBugList();
    public void handleShowHelpAbout();
    public void handleCascadewindows();
        public void handleMessageEntered(Object obj);
    public void handleTilewindows();
// added DSZ 3/29/99 - THIS SHOULD HAVE BEEN DONE BY WHOEVER IMPLEMENTED THE FOLLOWING!
public void handlesaveAllSettings();
public void handleReloadSettings();
// added DSZ 4/12/99 - this really should have been done the first time!!
public void handleChannelProperties();
}

//public static final int LOGO_HEIGHT = 29;
protected static final boolean LOCAL_DEBUG = false;
protected boolean showHints = false; // !!! ALERT - DSZ - CLEAN THIS UP WITH APPROPRIATE
ACCESSORS
//private ChannelwindowController controller;
protected MenuView menuView;
protected ChannelwindowViewMenuToolbarSet toolbarSet;

protected Containerview container;
protected MessageDisplayTextView contentView;
protected LockingScrollGroup contentScroll;
protected ScrollGroup entryScroll;
protected Splitview splitView;
protected MessageEntryTextView entryField;
protected ChannelSettings channelSettings;

// DSZ - the fields for the find dialog SHOULD NOT BE HERE!!!! It should be its
// own widget
// find dialog
protected ExternalWindow                                findDialog;
protected CutCopyPasteTextField        findTextField;
protected Button
findButton;
protected Button
findNextButton;
protected Button
doneButton;
protected Button
caseButton;

protected int margin = 2;
protected int menuToolbarHeight = 0;

protected boolean dockedState = false;
protected boolean startNewSearch = true;
protected boolean displayable = false;
protected boolean usingFinder = false;
private boolean hasCustomPanel;

protected PopupMenuProvider curPopupProvider = null;
protected PopupMenuConsumer curPopupConsumer = this;
protected String linkString = null;
protected static final String STOCK_CONTEXT = "STOCK_CONTEXT";

```

```

                                ChannelWindowView.java
protected CustomLinkHandler customLinkHandler;
protected CustomInputPanel customInputPanel;

private String channelName;
private User owningUser;

// constructor
public ChannelWindowView() {
    super();
    this.displayable = false;
    //setTarget(this);
    toolbarSet.setOwner(this);
    customLinkHandler = CustomLinkHandler.getSingleton();
    setChannelSettings(new ChannelSettings());
}

/**
 * Accessor method for ChannelWindowViewOwner
 */
public void setOwner(Owner own) {
    owner = (own instanceof ChannelWindowViewOwner) ? own : null;
}

/**
 * Accessor method for ChannelWindowViewOwner
 */
public Owner getOwner() {
    return this.owner;
}

/**
 * Accessor method for channel settings object
 */
public void setChannelSettings(ChannelSettings sets) {
    this.channelSettings = sets;
    setChannelName(getChannelName());
    contentView.setChannelSettings(sets);
    // apply these channel settings to the view
    applyChannelSettings();
}

/**
 * Accessor method for channel settings object
 */
public ChannelSettings getChannelSettings() {
    return this.channelSettings;
}

/**
 * This method should be called whenever the settings object for this channel
 * changes - it will update the display however necessary to reflect those
 * changes.
 * -- dsz
 */
public void applyChannelSettings() {
    // tell the toolbar that it is to be configured as "dock on"
    toolbarSet.setDockEnabled(true);
    // call all accessor methods to set new state
    setBackgroundColor(getBackgroundColor());
    setChannelName(getChannelName());
    setDockedState(getDockedState());
    setHasCustomPanel(getHasCustomPanel());
    setOwningUser(getOwningUser());
}

```

```

                                ChannelWindowView.java
setShowChannelColumn(getShowChannelColumn());
setShowCustomPanel(getShowCustomPanel());
setShowInputField(getShowInputField());
//setShowInputFieldEnabled(getShowInputFieldEnabled());
setShowJoinPart(getShowJoinPart());
setShowJoinPartEnabled(getShowJoinPartEnabled());
//setShowMenu(getShowMenu());
//setShowTextEditControl(getShowTextEditControl());
setShowTimeColumn(getShowTimeColumn());
setShowToolBar(getShowToolBar());
setShowUserColumn(getShowUserColumn());
setShowUserList(getShowUserList());

setShowUserListEnabled(getShowUserListEnabled());

// settings for different versions of interchange

toolbarSet.setShowBugListItem(Interchange.getInterchangeDataProvider().isBugListMenuShowing()
);

toolbarSet.setNewIRCEnabled(!Interchange.getInterchangeDataProvider().isRestrictedVersion());
toolbarSet.setShowBranding(Interchange.getInterchangeDataProvider().isBrandedVersion());

toolbarSet.setPostFileEnabled(Interchange.getInterchangeDataProvider().getFiletransferServerName() != null);
// user list must always show for external IC, set here instead of changing the get
functions
// to put all the version stuff in one place
// other things without accessors
toolbarSet.setChannelTypeBitmap(getChannelBitmap());
//contentview.applyChannelSettings();
// added DSZ
//setMakeOperatorEnabled(getMakeOperatorEnabled());
//setKickEnabled(getKickEnabled());

// Modified by raduload (9/22/98) and again by DSZ 4/13/99
// Disable the userlist toggle buttons with channels if running Client Interchange
// !!! ALERT - DSZ

//setShowUserListEnabled(!Interchange.getInterchangeDataProvider().isRestrictedVersion());
//toolbarSet.setShowBranding(Interchange.getInterchangeDataProvider().isBrandedVersion());

if (getChannelSettings().getWindowRaiseOnAlertOnly())
    setWindowRaiseState(ChannelWindowViewMenuToolBarSet.BUTTON_COMMAND_WINDOW_ON_ALERT);
else if (getChannelSettings().getWindowRaiseOn())
    setWindowRaiseState(ChannelWindowViewMenuToolBarSet.BUTTON_COMMAND_WINDOW_ON);
else
    setWindowRaiseState(ChannelWindowViewMenuToolBarSet.BUTTON_COMMAND_WINDOW_OFF);
if (getChannelSettings().getSoundOn())
    setSoundPlayState(ChannelWindowViewMenuToolBarSet.BUTTON_COMMAND_SOUND_ON);
else if (getChannelSettings().getSoundOnAlertOnly())
    setSoundPlayState(ChannelWindowViewMenuToolBarSet.BUTTON_COMMAND_SOUND_ON_ALERT);
else
    setSoundPlayState(ChannelWindowViewMenuToolBarSet.BUTTON_COMMAND_SOUND_OFF);
}

public void setShowJoinPartEnabled(boolean enabled) {
    toolbarSet.setShowJoinPartEnabled(enabled);
}

public boolean getShowJoinPartEnabled() {
    return toolbarSet.getShowJoinPartEnabled();
}

public void setShowJoinPart(boolean show) {
    getChannelSettings().setShowJoinLeave(new Boolean(show));
}

```

```

                                ChannelWindowView.java
    toolbarSet.setShowJoinPartState(show);
}

    public boolean getShowJoinPart() {
        return getChannelSettings().getShowJoinLeave().booleanValue();
    }

    public void setWindowRaiseState(String state) {
        // !!! ALERT - DSZ - how to set into channelSettigns object?
        toolbarSet.setWindowState(state);
    }

    public void setSoundPlayState(String state) {
        // !!! ALERT - DSZ - how to set into channelSettigns object?
        toolbarSet.setSoundState(state);
    }

    public void setMakeOperatorEnabled(boolean show) {
        toolbarSet.setMakeOperatorEnabled(show);
    }

    public boolean getMakeOperatorEnabled() {
        if (getOwningUser() != null)
            return getOwningUser().isPrivileged();
        return false;
    }

    public void setKickEnabled(boolean show) {
        toolbarSet.setKickEnabled(show);
    }

    public boolean getKickEnabled() {
        if (getOwningUser() != null)
            return getOwningUser().isPrivileged();
        return false;
    }

    public void setChannelPropertiesEnabled(boolean enabled) {
        toolbarSet.setChannelPropertiesEnabled(enabled);
    }

    public boolean getChannelPropertiesEnabled() {
        if (getOwningUser() != null)
            return getOwningUser().isPrivileged();
        return false;
    }

    protected MenuView getMenuView() {
        return toolbarSet.getMenu().getMenuView();
    }

    protected Toolbar getToolbar() {
        return toolbarSet.getToolbar().getToolbar();
    }

    public void setChannelName(String name) {
        if (name == null)
            return;

```

ChannelwindowView.java

```

channelName = name;
getChannelSettings().setName(name);
toolbarSet.setChannelName(name);
}

public String getChannelName() {
    return channelName;
}

public abstract Bitmap getChannelBitmap();

public void setBackgroundColor(Color newBackgroundColor) {
    getChannelSettings().setBGColor(newBackgroundColor);
    contentView.setBackgroundColor(newBackgroundColor);
    contentScroll.setBackgroundColor(newBackgroundColor);
}

public color getBackgroundColor() {
    return getChannelSettings().getBGColor();
}

// DSZ - controller should be setting this flag.
// !!! ALERT - DSZ - this seems a little strange to me.
public void setHasCustomPanel(boolean has) {
    hasCustomPanel = has;
    toolbarSet.setCustomInputPanelEnabled(has);
}

public boolean getHasCustomPanel() {
    return hasCustomPanel;
}

// added 4/6/99 by DSZ
public void setShowCustomPanel(boolean show) {
    // if I don't have a panel and my settings have show to false, then do nothing.
    otherwise
    // make sure the states are consistent
    if (!getHasCustomPanel()) {
        toolbarSet.setShowCustomPanelState(false);
        if (!getShowCustomPanel())
            return;
        // !!! ALERT - DSZ - change debug level
        System.err.println("Channel " + getChannelName() + " does not have a custom input panel
associated with it");
        getChannelSettings().setShowCustomInput(new Boolean(false));
        return;
    }
    if (show) {
        if (getCustomInputPanel() == null)
            customInputPanel = createCustomInputPanel();
        if (getCustomInputPanel() != null) {
            getCustomInputPanel().setOwner(this);
            container.addSubview(getCustomInputPanel());
        } else {
            // for some reason, custom input panel isn't available.
            Toolkit.getDefaultToolkit().beep();
            System.err.println("Channel " + getChannelName() + " was unable to load/display
custom input panel");
        }
    } else {
        if (getCustomInputPanel() != null)
            getCustomInputPanel().removeFromSuperview();
    }
}

```

ChannelWindowView.java

```

    }
    getChannelSettings().setShowCustomInput(new Boolean(show));
    toolbarSet.setShowCustomPanelState(show);
    setupFocus();
    resetSizes();
}

public boolean getShowCustomPanel() {
    return getChannelSettings().getShowCustomInput().booleanValue();
}

// added 4/6/99 by DSZ
protected CustomInputPanel createCustomInputPanel() {
    CustomInputPanel panel = null;
    Class customPanelClass = null;
    // dynamically load the panel class used by that channel
    try {
        customPanelClass =
            Class.forName((String)Interchange.panelPreferences.get(getChannelName().toUpperCase()));
    }
    catch (ClassNotFoundException e) {
        System.out.println("ClassNotFoundException : " +
            Interchange.panelPreferences.get(getChannelName().toUpperCase()));
        e.printStackTrace();
    }
    // create an instance of the custom panel class
    if (customPanelClass != null) {
        try {
            panel = (CustomInputPanel)customPanelClass.newInstance();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    return panel;
}

protected CustomInputPanel getCustomInputPanel() {
    return customInputPanel;
}

public void setShowUserListEnabled(boolean enable) {
    if (Interchange.getInterchangeDataProvider().isRestrictedVersion())
        enable = false;
    toolbarSet.setUserListEnabled(enable);
}

// allow subclasses to determine whether or not user list should be enabled
public abstract boolean getShowUserListEnabled();

public void setShowUserList(boolean show) {
    if (Interchange.getInterchangeDataProvider().isRestrictedVersion())
        show = true; // DSZ - force if restricted version
    getChannelSettings().setShowUserList(new Boolean(show));
    toolbarSet.setUserListShowing(show);
    resetSizes();
    setDirty(true);
}

public boolean getShowUserList() {
    return getChannelSettings().getShowUserList().booleanValue();
}

```

ChannelwindowView.java

```
// DSZ - this is ignored. we don't do anything with this anyway
//public void setShowMenu(boolean show) {
//    getChannelSettings().setShowMenu(new Boolean(show));
//}

// DSZ - this is ignored. we don't do anything with this anyway
//public boolean getShowMenu() {
//    return getChannelSettings().getShowMenu().booleanValue();
//}

public void setShowTimeColumn(boolean show) {
    toolbarSet.setShowTimeColumnState(show);
    // if (getChannelSettings().getShowTime().booleanValue() == show)
    //    return;
    getChannelSettings().setShowTime(show);
    if (contentView.showTime() != show) {
        contentView.setShowTime(show);
        contentView.reloadContent();
    }
    resetContentscroller();
}

public boolean getShowTimeColumn() {
    return getChannelSettings().getShowTime().booleanValue();
}

public void setShowUserColumn(boolean show) {
    toolbarSet.setShowUserColumnState(show);
    // if (getChannelSettings().getShowUser().booleanValue() == show)
    //    return;
    getChannelSettings().setShowUser(new Boolean(show));
    if (contentView.showUser() != show) {
        contentView.setShowUser(show);
        contentView.reloadContent();
    }
    resetContentscroller();
}

public boolean getShowUserColumn() {
    return getChannelSettings().getShowUser().booleanValue();
}

public void setShowChannelColumn(boolean show) {
    toolbarSet.setShowChannelColumnState(show);
    // if (getChannelSettings().getShowChannel().booleanValue() == show)
    //    return;
    getChannelSettings().setShowChannel(new Boolean(show));
    if (contentView.showChannel() != show) {
        contentView.setShowChannel(show);
        contentView.reloadContent();
    }
    resetContentscroller();
}

public boolean getShowChannelColumn() {
    return getChannelSettings().getShowChannel().booleanValue();
}

public void setShowInputField(boolean show) {
    toolbarSet.setShowInputFieldState(show);
    // if (getChannelSettings().getShowInputField().booleanValue() == show)
```


ChannelwindowView.java

```
// return;
getChannelSettings().setShowInputField(new Boolean(show));
if (show)
    splitView.showView2();
else
    splitView.hideview2();
resetSizes();
setupFocus();
}

public boolean getShowInputField() {
    return getChannelSettings().getShowInputField().booleanValue();
}

/*
public void setShowInputFieldEnabled(boolean enabled) {
    // !!! ALERT - DSZ - what determines this?
    toolbarSet.setShowInputFieldEnabled(enabled);
}

// override this to force input field to be disabled
public boolean getShowInputFieldEnabled() {
    // !!! ALERT - DSZ - what determines how this is enabled/disabled?
    return true;
    //return toolbarSet.getShowInputFieldEnabled();
}
*/

public void setShowToolbar(boolean show) {
    toolbarSet.setDisplayToolbar(show);
    getChannelSettings().setShowToolbar(new Boolean(show));
    resetSizes();
    setDirty(true);
}

public boolean getShowToolbar() {
    return getChannelSettings().getShowToolbar().booleanValue();
}

/*
public void setShowTextEditControl(boolean show) {
    getChannelSettings().setShowTextEdit(new Boolean(show));
    resetSizes();
}

public boolean getShowTextEditControl() {
    return getChannelSettings().getShowTextEdit().booleanValue();
}
*/

public void setDockedState(boolean value) {
    // !!! ALERT - DSZ
    showHints = value;
    dockedState = value;
    toolbarSet.setDockedState(dockedState);
}

public boolean getDockedState() {
    return dockedState;
}
```

ChannelWindowView.java

```
// !!! ALERT - DSZ
public void setHints(boolean value) {
    showHints = value;
}

public void setOwningUser(User usr) {
    owningUser = usr;
    toolbarSet.setUser(usr);
    if (usr == null)
        return;
    setMakeOperatorEnabled(getMakeOperatorEnabled());
    setKickEnabled(getKickEnabled());
    setChannelPropertiesEnabled(getChannelPropertiesEnabled());
}

public User getOwningUser() {
    return owningUser;
}

// derived class should override this method to
// return the width of the user list panel so that
// the super class can size the split view properly
protected int getUserListwidth() {
    return 0;
}

/**
 *      *      Overridden from CustomView. Called from the constructor when the
dgets *      need to be created. Note how here they are created and configured, but not
      *      positioned or sized.
      */
protected void createWidgets() {
    // create main container
    container = new ContainerView();
    container.setBorder(new EmptyBorder());
    // toolbar setup
    toolbarSet = new ChannelWindowViewMenuToolbarSet(null);
    toolbarSet.setTarget(this);
    //toolbarSet.config(this);
    addSubview(toolbarSet);
    // contentView setup
    contentView = new MessageDisplayTextView();
    contentView.setEditable(false);
    contentView.setSelectable(true);
    contentView.setOwner(this);
    contentScroll = new LockingScrollGroup();
    contentScroll.setVertScrollBarDisplay(ScrollGroup.ALWAYS_DISPLAY);
    //contentScroll.setBuffered(true);
    contentScroll.setContentView(contentView);
    contentView.moveTo(0,0);
    // message entry setup
    entryField = new MessageEntryTextView();
    entryField.setPopupMenuProvider(this);
    entryField.setFont(Font.fontNamed("SansSerif", Font.PLAIN, 11));
    entryField.setTarget(this);
    entryScroll = new ScrollGroup();
    entryScroll.setHashHorizScrollBar(false);
    entryScroll.setVertScrollBarDisplay(ScrollGroup.AS_NEEDED_DISPLAY);
    entryScroll.setBackgroundColor(Color.white);
    entryScroll.setMinSize(10, 16 + SMALL_MARGIN);
    entryScroll.setContentView(entryField);
    entryField.moveTo(0,0);
    //entryField.setFocusedView();
    // split view setup
}
```

```

splitview = new SplitView();
splitview.setOwner(this);
splitview.setShowDimple(true);
splitview.setView1(contentScroll);
    contentScroll.moveTo(0,0);
splitview.setView2(entryScroll);
entryScroll.moveTo(0,0);
container.addSubview(splitview);
splitview.moveTo(SMALL_MARGIN, SMALL_MARGIN);
// add the container
addSubview(container);
}

public void startHint(FloatingHintConsumer consumer, String context) {
    if (showHints)
        FloatingHintResources.getFloatingHintResources().startHint(consumer,
context);
}

    public void endHint(FloatingHintConsumer consumer) {
        if (showHints)
            FloatingHintResources.getFloatingHintResources().endHint(consumer);
    }

public void appendContent(Content content) {
    ((LockingScrollBar)contentScroll.vertScrollBar()).contentAppending();
    contentView.appendContent(content);
    // scroll down to the bottom
    resetContentsCroller();
}

public void resetContentsCroller() {
    // check to see if the user is using the finder. If they are then don't scroll to
the bottom
    if (!usingFinder &&
!((LockingScrollBar)contentScroll.vertScrollBar()).getLocked()) {
        contentScroll.scrollView().scrollRectToVisible(new Rect(0, contentView.height() - 1,
contentView.width(), contentView.height()));
    }
}

    public void clearContent() {
        contentView.clearBuffer();
    }

public void refreshContent() {
    contentView.refreshContent();
}

/*
 * This temporarily solves the window pop-to-front problem. When a window pops to
 * front, the focus is taken away from the previously focused window and is put in
 * the text entry field of the popped window. This may be dangerous, allowing a user
 * type mistype information into the wrong window. Solution: just don't focus on the
 * text field in this case only.
 */
public void setupFocusForWindowRaise() {
    if (LOCAL_DEBUG || util.Globals.DEBUG) {
        System.out.println("Setting Focus for Window Raise");
        System.out.println("Setting the ContentView as the focused view");
    }
    RootView rv = rootView();
    if (rv != null) {

```

```

        Channelwindowview.java
        rv.setDefaultSelectedView(contentView);
        rv.selectView(contentView, true);
        rv.setFocusedView(contentView);
    } else {
        contentView.setFocusedView();
    }
}

/**
 *      Implemented from CustomView. The "owner" of this view can call this
 *      method to "start" the default focusing.
 */
public void setupFocus() {
    RootView rv = rootView();
    if (rv != null) {
        // Modified by raduload (9/28/98) - if the input field is not enabled, do not allow it
        focus if (rv.focusedView() == contentView || !getShowInputField()) {
            if (LOCAL_DEBUG || util.Globals.DEBUG) {
                System.out.println("ContentView is the focused view");
            }
            return;
        }
        if (LOCAL_DEBUG || util.Globals.DEBUG) {
            System.out.println("ContentView is NOT the focused view");
            System.out.println("Setting the EntryField as the focused view");
        }
        rv.setDefaultSelectedView(entryField);
        rv.selectView(entryField, true);
        rv.setFocusedView(entryField);
    } else {
        entryField.setFocusedView();
    }
}

public void setDisplayable(boolean isDisplayable) {
    displayable = isDisplayable;
}

/**
 *      Overridden from CustomView. Called when this view needs to be resized.
 *      All widget layout and sizing should be performed in this method.
 */
public void resetSizes() {
    if (!displayable) return;
    MenuView menuView = toolbarSet.getMenu().getMenuView(); // get the MenuView
    //toolbarSet.setDockState(getDockedState()); // sets up dock/undock button
    if (!getDockedState()) {
        toolbarSet.sizeToMinSize();
        toolbarSet.moveTo(startPosX, startPosY);
        if (toolbarSet.getMenu().windowShowHideToolbarMI.state()) {
            toolbarSet.addSubview(toolbarSet.getToolbar().getToolbar());
            toolbarSet.sizeTo(toolbarSet.getToolbar().getToolbar().height() +
                menuView.height());
            menuToolbarHeight = toolbarSet.getToolbar().getToolbar().height() +
                menuView.height();
        }
        else {
            toolbarSet.getToolbar().getToolbar().removeFromSuperview();
            toolbarSet.sizeTo(toolbarSet.getToolbar().getToolbar().height());
            menuToolbarHeight = menuView.height();
        }
    }
    else {
        toolbarSet.moveTo(startPosX, startPosY);
        toolbarSet.sizeTo(0,0);
        toolbarSet.getToolbar().getToolbar().removeFromSuperview();
    }
}

```

```

    menuToolBarHeight = 1;
}
// size and layout the container
container.moveTo(startPosX, menuToolBarHeight - 1);
container.setSize(totWidth, totHeight - menuToolBarHeight);
if (getShowCustomPanel() && (getCustomInputPanel() != null)) {
    getCustomInputPanel().setSize(totWidth - 2*SMALL_MARGIN,
getCustomInputPanel().PREFERRED_HEIGHT - 2*SMALL_MARGIN);
    getCustomInputPanel().moveTo(SMALL_MARGIN, totHeight - menuToolBarHeight -
getCustomInputPanel().PREFERRED_HEIGHT + SMALL_MARGIN);
}
// added 4/6/99 by DSZ for custom input panel
int ht = container.height() - SMALL_MARGIN*2 - container.border().heightMargin();
if (getShowCustomPanel() && (getCustomInputPanel() != null))
    ht -= getCustomInputPanel().PREFERRED_HEIGHT + SMALL_MARGIN;
    if(!getShowInputField()) {
        contentScroll.setSize(container.width() - SMALL_MARGIN*2 -
container.border().widthMargin(), ht);
        contentScroll.moveTo(SMALL_MARGIN, SMALL_MARGIN);
    }
// record scroll positions
float contentScrollPos = contentScroll.getVerticalScrollBar().getScrollPercent();
float entryScrollPos = entryScroll.getVerticalScrollBar().getScrollPercent();
//updateChannelMenu(); DSZ - this certainly doesn't belong here - the name doesn't
change.
toolbarSet.setDockedState(getDockedState()); // sets up dock/undock button
int userListX;
if (getShowUserList())
    userListX = totWidth - getUserListWidth() - SMALL_MARGIN;
else
    userListX = totWidth - SMALL_MARGIN;
// resize the split view
if (getShowCustomPanel() && (getCustomInputPanel() != null))
    splitView.setSize(userListX - SMALL_MARGIN, totHeight - menuToolBarHeight -
2*SMALL_MARGIN - customInputPanel.PREFERRED_HEIGHT);
else
    splitView.setSize(userListX - SMALL_MARGIN, totHeight - menuToolBarHeight -
2*SMALL_MARGIN);

// resize the contentview and entryField
Rect contentBounds = contentScroll.getScrollView().getBounds();
contentView.setSizeToMinSize();
contentView.setSizeTo(contentBounds.width, contentBounds.height);
// resize the entryField
contentBounds = entryScroll.getScrollView().getBounds();
entryField.setSizeToMinSize();
entryField.setSizeTo(contentBounds.width, contentBounds.height);
// reset scroll to its original position
contentScroll.getVerticalScrollBar().setScrollPercent(contentScrollPos);
entryScroll.getVerticalScrollBar().setScrollPercent(entryScrollPos);
// scroll down to end of content
resetContentScroller();
// changed, dsz
//if (getShowTextEditControl()) {
//if (showTextEditControl) {
//    entryScroll.setSize(0, 0);
//    entryScroll.moveTo(-10, -10);
//}
if (splitView.getView2() != null) {
    splitView.setSplitBarPos(splitView.getHeight() -
getChannelSettings().getSplitterPosition().intValue() - splitView.getSplitBarSize());
} else {
    splitView.setSplitBarPos(splitView.getHeight()-splitView.getSplitBarSize());
}
}

/**
 * overridden from View to indicate when I want a popup menu. Remember
 * that my subviews will get their own mouseDown and this one will not be called.

```

```

*/
public boolean mouseDown(MouseEvent e) {
    if (e.modifiers() == java.awt.Event.META_MASK) {
        requestPopupMenu(this, e, null);
        return false;
    } else {
        return super.mouseDown(e);
    }
}

public void performCommand(String command, Object _object) {
    if (_object == findButton)
        handleFind();
    else if (_object == findNextButton)
        handleFindNext();
    else if (_object == doneButton) {
        handleCancelFind();
        findDialog.hide();
    }
    else if (command.equals(MessageEntryTextView.TEXTVIEW_STRING_ENTERED))
        handleMessageEntered(_object);
    else if (customLinkHandler.isCustomCommand(command))
        customLinkHandler.handleLink(command, linkString);
}

protected void configureFindDialog() {
    if (findDialog != null)
        return;

    // create the external window
    findDialog = new Externalwindow();
    // position the window
    findDialog.setSize(325, 80);
    WindowManager.GetWindowManager().centerWindow(findDialog);
    //findDialog.moveTo(200, 200);
    // configure the window
    findDialog.panel().rootView().setColor(Color.lightGray);
    findDialog.setResizable(false);
    findDialog.setTitle("Find");
    // added DSZ 4.28.99 - if dialog is closed with "x" then execute cancel
    findDialog.setOwner(new WindowOwner() {
        public boolean windowWillShow(Window awindow) { return true; }
        public void windowDidShow(Window awindow) {}
        public boolean windowWillHide(Window awindow) {
            handleCancelFind();
            return true;
        }
    });
    public void windowDidHide(Window awindow) {}
    public void windowDidBecomeMain(Window awindow) {}
    public void windowDidResignMain(Window awindow) {}
    public void windowWillSizeBy(Window awindow, Size deltaSize) {}
    });
    // create the find label
    Label findLabel = new Label("Find what:", Font.fontNamed("Dialog", Font.PLAIN, 12));
    findLabel.setSize(findLabel.getWidth(), 20);
    findDialog.addSubview(findLabel);
    findLabel.moveTo(MARGIN, MARGIN);
    // create the find text field
    findTextField = new CutCopyPasteTextField();
    findTextField.setOwner(this);
    findDialog.addSubview(findTextField);
    findTextField.moveTo(findLabel.x() + findLabel.width() + MARGIN, findLabel.y());
    findTextField.setSize(findDialog.contentSize().width - findTextField.x(),
        findLabel.height());
    // create find next button
    findNextButton = new Button();
    findNextButton.setTitle("Find Next");
    findNextButton.setCommand("findNext");
}

```

ChannelWindowView.java

```

        findNextButton.setTarget(this);
        findNextButton.setSize(findNextButton.minSize().width + MARGIN,
findNextButton.minSize().height + MARGIN);
        findDialog.addSubview(findNextButton);
        // create find button
        findButton = new Button();
        findButton.setTitle("Find");
        findButton.setCommand("findNew");
        findButton.setTarget(this);
        findButton.setSize(findNextButton.minSize().width + MARGIN,
findNextButton.minSize().height + MARGIN);
        findDialog.addSubview(findButton);
        // create cancel button
        doneButton = new Button();
        doneButton.setTitle("Done");
        doneButton.setCommand("doneFind");
        doneButton.setTarget(this);
        doneButton.setSize(findNextButton.width(), findNextButton.height());
        findDialog.addSubview(doneButton);
        doneButton.moveTo(findDialog.contentSize().width - doneButton.width(),
findDialog.contentSize().height - doneButton.height());
        // move the find next and find buttons left of the cancel button
        findNextButton.moveTo(doneButton.x() - findNextButton.width() - MARGIN, doneButton.y());
        findButton.moveTo(findNextButton.x() - findNextButton.width() - MARGIN, doneButton.y());
        // create case sensitive button
        caseButton = Button.createCheckButton(0,0,0,0);
        caseButton.setFont(findLabel.font());
        caseButton.setTitle("Ignore Case");
        caseButton.setSizeToMinSize();
        caseButton.setState(true);
        findDialog.addSubview(caseButton);
        caseButton.moveTo(findButton.x() - caseButton.width() - MARGIN, doneButton.y());
    }
    //*****

protected void showFindDialog() {
    if (!findDialog.isVisible()) {
        findDialog.showModally();
        findDialog.panel().rootView().setDefaultSelectedView(findTextField);
        findDialog.panel().rootView().setFocusedView(findTextField);
        findTextField.selectText();
    }
}

public void handleToggleJoinPart() {
    setShowJoinPart(!getShowJoinPart());
}

public void handleNonSpecificPrivate(){
    if (getOwner() != null)
        ((ChannelWindowViewOwner) getOwner()).handleNonSpecificPrivate();
}

public void handleNonSpecificwhois() {
    if (getOwner() != null)
        ((ChannelWindowViewOwner) getOwner()).handleNonSpecificwhois();
}

// implemented by those subclasses who care.
public abstract void handleMakeOperator();
public abstract void handleKick();
public abstract void handleInvite(Channel channel);

public void handlechannelPrefs() {

```

ChannelwindowView.java

```
if(getOwner() != null)
    ((ChannelwindowViewOwner) getOwner()).handleChannelPrefs();
}

public void handleChannelPrefsDefault(){
    if(getOwner() != null)
        ((ChannelwindowViewOwner) getOwner()).handleChannelPrefsDefault();
}

    public void handleChannelPrefsGeneral(){
        if(getOwner() != null)
            ((ChannelwindowViewOwner) getOwner()).handleChannelPrefsGeneral();
    }

public void handleDock() {
    if(getOwner() != null)
        ((ChannelwindowViewOwner) getOwner()).handleDock();
}

public void handleUnDock() {
    if(getOwner() != null)
        ((ChannelwindowViewOwner) getOwner()).handleUnDock();
    applyChannelSettings();
}

public void handleChannelClose() {
    if(getOwner() != null)
        ((ChannelwindowViewOwner) getOwner()).handleChannelClose();
}

public void handleChannelClear() {
    if(getOwner() != null)
        ((ChannelwindowViewOwner) getOwner()).handleChannelClear();
}

public void handlePostFile() {
    if(getOwner() != null)
        ((ChannelwindowViewOwner) getOwner()).handlePostFile();
}

public void handleGetChannel(String channelName) {
    if(getOwner() != null)
        ((ChannelwindowViewOwner) getOwner()).handleGetChannel(channelName);
}

public void handleShowChannelListwindow() {
    if(getOwner() != null)
        ((ChannelwindowViewOwner) getOwner()).handleShowChannelListwindow();
}

public void handleDockAll() {
    if(getOwner() != null)
        ((ChannelwindowViewOwner) getOwner()).handleDockAll();
}

public void handleQuitApplication() {
    if(getOwner() != null)
        ((ChannelwindowViewOwner) getOwner()).handleQuitApplication();
}
```


ChannelwindowView.java

```
public void handleShowHelpContents() {
    if(getOwner() != null)
        ((ChannelWindowViewOwner) getOwner()).handleShowHelpContents();
}

public void handleShowHelpBugList() {
    if(getOwner() != null)
        ((ChannelWindowViewOwner) getOwner()).handleShowHelpBugList();
}

public void handleShowHelpAbout() {
    if(getOwner() != null)
        ((ChannelWindowViewOwner) getOwner()).handleShowHelpAbout();
}

protected void handleSave(){
    if(getOwner() != null)
        ((ChannelWindowViewOwner) getOwner()).handleSaveAllSettings();
}

protected void handleReload(){
    if(getOwner() != null)
        ((ChannelWindowViewOwner) getOwner()).handleReloadSettings();
}

public void handleCascadewindows() {
    if(getOwner() != null)
        ((ChannelWindowViewOwner) getOwner()).handleCascadewindows();
}

    protected void handleMessageEntered(Object obj) {
        if(getOwner() != null)
            ((ChannelWindowViewOwner) getOwner()).handleMessageEntered(obj);
    }

public void handleTilewindows() {
    if(getOwner() != null)
        ((ChannelWindowViewOwner) getOwner()).handleTilewindows();
}

protected void handleShowFindDialog() {
    configureFindDialog();
    showFindDialog();
    startNewSearch = true;
}

protected void handleFind() {
    startNewSearch = true;
    contentView.resetFindSettings();
    handleFindNext();
}

protected void handleFindNext() {
    usingFinder = true;
    if (findDialog != null) {
        Range resultRange = null;
        if (startNewSearch) {
            if ((resultRange = contentView.find(findTextField.stringValue(), caseButton.state()))
```

```

== null)
    java.awt.Toolkit.getDefaultToolkit().beep();
} else {
    if ((resultRange = contentView.findAgain(caseButton.state())) == null)
        java.awt.Toolkit.getDefaultToolkit().beep();
    }
    if (resultRange != null) {
        // scroll to the match and select it
        rootView().setFocusedView(contentView);
        contentView.scrollRangeToVisible(resultRange);
        contentView.selectRange(resultRange);
    }
}
startNewSearch = false;
}

protected void handleCancelFind() {
    //if (findDialog != null)
    //    findDialog.hide();
    contentView.resetFindSettings();
    usingFinder = false;
    resetContentScroller();
}

protected void handleCut() {
    if (rootView().focusedView() == entryField)
        entryField.performCommand("cut", this);
    else if ((rootView().focusedView() != null) && (rootView().focusedView() instanceof
TextField))
        ((TextField) rootView().focusedView()).cut();
    else
        handleCopy();
}

protected void handleCopy() {
    if (rootView().focusedView() == entryField)
        entryField.performCommand("copy", this);
    else if ((rootView().focusedView() != null) && (rootView().focusedView() instanceof
TextField))
        //Application.setClipboardText(((TextField)
rootView().focusedView()).selectedStringValue());
        ((TextField) rootView().focusedView()).copy();
    else if ((rootView().focusedView() != null) && (rootView().focusedView() == contentView))
        Application.setClipboardText(contentView.getSelectedText());
}

protected void handlePaste() {
    if (rootView().focusedView() == entryField)
        entryField.performCommand("paste", this);
    else if ((rootView().focusedView() != null) && (rootView().focusedView() instanceof
TextField))
        ((TextField) rootView().focusedView()).paste();
}

protected void handleChannelProperties() {
    if (getOwner() != null)
        ((ChannelWindowViewOwner) getOwner()).handleChannelProperties();
}

public void handleToolbarToggle() {
    setShowToolbar(!getShowToolbar());
}

```

ChannelWindowView.java

```
public void handleToggleSoundOn() {
    getChannelSettings().setSoundOn(true);
}

public void handleToggleSoundAlert() {
    getChannelSettings().setSoundOnAlertOnly();
}

public void handleToggleSoundOff() {
    getChannelSettings().setSoundOn(false);
}

public void handleToggleWindowOn() {
    getChannelSettings().setWindowRaiseOn(true);
}

public void handleToggleWindowAlert() {
    getChannelSettings().setWindowRaiseOnAlertOnly();
}

public void handleToggleWindowOff() {
    getChannelSettings().setWindowRaiseOn(false);
}

protected void handleTimeColumnToggle() {
    setShowTimeColumn(!getShowTimeColumn());
}

protected void handleChannelColumnToggle() {
    setShowChannelColumn(!getShowChannelColumn());
}

protected void handleUserColumnToggle() {
    setShowUserColumn(!getShowUserColumn());
}

protected void handleInputFieldToggle() {
    setShowInputField(!getShowInputField());
}

public void handlePrint() {
    contentView.print();
}

public void handlePrintPreview() {
    contentView.printPreview();
}

public void handleToggleUserList(){
    // this will be called when the toolbarset fires the event - from menu or toolbar - play
    sound here
    if (getShowUserList()) {
        ((Interchange)
netscape.application.Application.application()).playSound(ChannelResources.USER_LIST_OPEN_SOUND);
    } else {
        ((Interchange)
netscape.application.Application.application()).playSound(ChannelResources.USER_LIST_CLOSE_SOUND);
    }
}
```

```

UND);
}
}
setShowUserList(!getShowUserList());
}

    public void handleSetupToCreateNewChannel(String type) {
    if (getOwner() != null)
        ((ChannelWindowViewOwner) getOwner()).handleShowCreationView(type);
    }

// added 4/6/99 by DSZ
public void handleToggleCustomPanel() {
    setShowCustomPanel(!getShowCustomPanel());
}

public void messageEntered(String aMessage) {
    handleMessageEntered(aMessage);
}

// added addMessage to allow SIP to post message to its own
// channel window only (Bug0414 - Jiefei)
public void addMessage(String aMessage) {
    Content content = new Content();
    content.setAuthorType(Content.INTERCHANGE);
    content.setAuthor("*****");
    content.setChannel(getChannelName());
    content.setContent(aMessage);
    appendContent(content);
}

    public void setStatusMessage(String str,float f) {}

public void setStatusMessage(String str) {}

//
// ----- GeneralMenuToolbarSetOwner implementation
//
public void menuToolbarSetDidSelectDockAll(GeneralMenuToolbarSet menuSet) {
    handleDockAll();
}

public void menuToolbarSetDidSelectExitApplication(GeneralMenuToolbarSet menuSet){
    handleQuitApplication();
}

public void menuToolbarSetDidSelectPrivate(GeneralMenuToolbarSet menuSet) {
    handleNonSpecificPrivate();
}

public void menuToolbarSetDidSelectWhois(GeneralMenuToolbarSet menuSet) {
    handleNonSpecificWhois();
}

public void menuToolbarSetDidSelectChannelPrefs(ChannelWindowViewMenuToolbarSet menuSet) {
    handleChannelPrefs();
}

public void menuToolbarSetDidSelectHelpContents(GeneralMenuToolbarSet menuSet) {
    handleShowHelpContents();
}

```

ChannelWindowView.java

```

}

public void menuToolBarSetDidSelectHelpAbout(GeneralMenuToolBarSet menuSet) {
    handleShowHelpAbout();
}

    public void menuToolBarSetDidSelectHelpBug(GeneralMenuToolBarSet menuSet) {
        handleShowHelpBugList();
    }

public void menuToolBarSetDidSelectWindowCascade(GeneralMenuToolBarSet menuSet) {
    handleCascadewindows();
}

public void menuToolBarSetDidSelectWindowTile(GeneralMenuToolBarSet menuSet) {
    handleTilewindows();
}

public void menuToolBarSetDidSelectWindowToggleToolBar(GeneralMenuToolBarSet menuSet) {
    handleToolBarToggle();
}

public void menuToolBarSetDidSelectPostFile(ChannelWindowViewMenuToolBarSet menuSet) {
    handlePostFile();
}

    public void menuToolBarSetDidSelectChannelPrefsDefault(ChannelWindowViewMenuToolBarSet
menuSet){
        handleChannelPrefsDefault();
    }

        public void
menuToolBarSetDidSelectChannelPrefsGeneral(ChannelWindowViewMenuToolBarSet menuSet){
            handleChannelPrefsGeneral();
        }

public void menuToolBarSetDidSelectSaveAllSettings(GeneralMenuToolBarSet menuSet) {
    handleSave();
}

public void menuToolBarSetDidSelectReloadSettings(GeneralMenuToolBarSet menuSet) {
    handleReload();
}

    public void menuToolBarSetDidSelectNewChannel(GeneralMenuToolBarSet menuSet, String
type) {
        handleSetupToCreateNewChannel(type);
    }

    //
    // ----- ChannelWindowViewMenuToolBarSetOwner implementation
    //
    public void menuToolBarSetDidSelectToggleShowJoinPart(ChannelWindowViewMenuToolBarSet
menuSet) {
        handleToggleJoinPart();
    }

```

ChannelWindowView.java

```
public void menuToolBarSetDidSelectCut(ChannelWindowViewMenuToolBarSet menuSet) {
    handleCut();
}

public void menuToolBarSetDidSelectCopy(ChannelWindowViewMenuToolBarSet menuSet) {
    handleCopy();
}

public void menuToolBarSetDidSelectPaste(ChannelWindowViewMenuToolBarSet menuSet) {
    handlePaste();
}

public void menuToolBarSetDidSelectDockChannel(ChannelWindowViewMenuToolBarSet menuSet) {
    handleDock();
}

public void menuToolBarSetDidSelectUndockChannel(ChannelWindowViewMenuToolBarSet menuSet) {
    handleUndock();
}

public void menuToolBarSetDidSelectCloseChannel(ChannelWindowViewMenuToolBarSet menuSet) {
    handleChannelClose();
}

public void menuToolBarSetDidSelectPrintChannel(ChannelWindowViewMenuToolBarSet menuSet) {
    handlePrint();
}

public void menuToolBarSetDidSelectPrintPreviewChannel(ChannelWindowViewMenuToolBarSet
menuSet) {
    handlePrintPreview();
}

public void menuToolBarSetDidSelectClearChannelHistory(ChannelWindowViewMenuToolBarSet
menuSet) {
    handleChannelClear();
}

public void menuToolBarSetDidSelectChannelProperties(ChannelWindowViewMenuToolBarSet
menuSet) {
    handleChannelProperties();
}

public void menuToolBarSetDidSelectOp(ChannelWindowViewMenuToolBarSet menuSet) {
    handleMakeOperator();
}

public void menuToolBarSetDidSelectKick(ChannelWindowViewMenuToolBarSet menuSet) {
    handleKick();
}

public void menuToolBarSetDidSelectSpecificPrivate(ChannelWindowViewMenuToolBarSet menuSet)
{
    handleNonSpecificPrivate();
}

public void menuToolBarSetDidSelectSpecificWhois(ChannelWindowViewMenuToolBarSet menuSet) {
```

```

    handleNonSpecificWhois();
}

    public void menuToolBarSetDidSelectToggleShowTimeColumn(ChannelWindowViewMenuToolBarSet
menuSet) {
    handleTimeColumnToggle();
}

    public void menuToolBarSetDidSelectToggleShowChannelColumn(ChannelWindowViewMenuToolBarSet
menuSet) {
    handleChannelColumnToggle();
}

    public void menuToolBarSetDidSelectToggleShowUserColumn(ChannelWindowViewMenuToolBarSet
menuSet) {
    handleUserColumnToggle();
}

    public void menuToolBarSetDidSelectToggleShowInputField(ChannelWindowViewMenuToolBarSet
menuSet) {
    handleInputFieldToggle();
}

    public void menuToolBarSetDidSelectToggleShowUserList(ChannelWindowViewMenuToolBarSet
menuSet) {
    handleToggleUserList();
}

    public void menuToolBarSetDidSelectToggleShowCustomPanel(ChannelWindowViewMenuToolBarSet
menuSet) {
    handleToggleCustomPanel();
}

    public void menuToolBarSetDidSelectSoundOff(ChannelWindowViewMenuToolBarSet menuSet) {
    handleToggleSoundOff();
}

    public void menuToolBarSetDidSelectSoundOn(ChannelWindowViewMenuToolBarSet menuSet) {
    handleToggleSoundOn();
}

    public void menuToolBarSetDidSelectSoundOnAlert(ChannelWindowViewMenuToolBarSet menuSet) {
    handleToggleSoundAlert();
}

    public void menuToolBarSetDidSelectWindowOff(ChannelWindowViewMenuToolBarSet menuSet) {
    handleToggleWindowOff();
}

    public void menuToolBarSetDidSelectWindowOn(ChannelWindowViewMenuToolBarSet menuSet) {
    handleToggleWindowOn();
}

    public void menuToolBarSetDidSelectWindowOnAlert(ChannelWindowViewMenuToolBarSet menuSet) {
    handleToggleWindowAlert();
}

```

```

                                ChannelWindowView.java
public void menuToolBarSetDidSelectFindContents(ChannelWindowViewMenuToolBarSet menuSet) {
    handleShowFindDialog();
}

public void menuToolBarSetDidSelectShowChannelManager(ChannelWindowViewMenuToolBarSet
menuSet) {
    handleShowChannelListWindow();
}

    // SplitViewOwner interface
    public boolean splitViewWillStartDragging(SplitView split) {
return true;
    }

    public void splitViewIsDragging(SplitView split) {};

    public void splitViewDidEndDragging(SplitView split) {
        getChannelSettings().setSplitterPosition(new Integer(splitView.getView2Size()));
    }

    // TextViewOwner interface
    public void attributesDidChange(TextView tv, Range r) {}

    public void attributesWillChange(TextView tv, Range r) {}

public void linkWasSelected(TextView tv, Range r, String linkString) {
    try {
        int urlIdx;
        if ((urlIdx = linkString.toLowerCase().indexOf(MessageDisplayTextView.URL_TAG)) > -1
) {
            // BUG0281. Handle the case #abchttp://def. GZ 7/5/99
            System.out.println("Opening " + linkString.substring(urlIdx));
            webUtilities.showDocument(linkString.substring(urlIdx));
        } else if ( linkString.toLowerCase().indexOf(MessageDisplayTextView.URLS_TAG) > -1 )
{
            int urlsIdx = linkString.toLowerCase().indexOf(MessageDisplayTextView.URLS_TAG);
            System.out.println("Opening " + linkString.substring(urlsIdx));
            webUtilities.showDocument(linkString.substring(urlsIdx));
        } else if ( linkString.toLowerCase().indexOf(MessageDisplayTextView.FTP_TAG) > -1 ) {
            int ftpIdx = linkString.toLowerCase().indexOf(MessageDisplayTextView.FTP_TAG);
            System.out.println("Opening " + linkString.substring(ftpIdx));
            webUtilities.showDocument(linkString.substring(ftpIdx));
        } else if ( linkString.indexOf(MessageDisplayTextView.IRC_TAG) > -1 ) {
            System.out.println("Opening channel " + linkString);
            //controller.handleGetChannel(linkString);
            if (getOwner() != null)
                ((ChannelWindowViewOwner) getOwner()).handleGetChannel(linkString);
        } else if ( linkString.startsWith(MessageDisplayTextView.STOCK_BEGIN_TAG) ) {
            // save the string for later use
            linkString = linkString.substring(1, linkString.length() - 1);
            // right click, bring up menu
            if(contentView.isRightClick())
                requestPopupMenu(this, contentView.getMouseEvent(), STOCK_CONTEXT);
            // left click default to kaleidoscope
            else
                customLinkHandler.handleKScopeCall(linkString);
        } else {
            if(LOCAL_DEBUG || util.Globals.DEBUG) {
                System.out.println("Opening private " + linkString);
            }
            if (getOwner() != null)
                ((ChannelWindowViewOwner) getOwner()).handleSpecificPrivate(new User(linkString));
            //controller.handleSpecificPrivate(new User(linkString));
        }
    }
}

```



```

    }
}
catch (Exception e) {
    System.out.println("Exception: " + e);
    e.printStackTrace();
}
}

// PopupMenuProvider and PopupMenuConsumer Interface
public void setPopupMenuProvider(PopupMenuProvider prov) {
    curPopupMenuProvider = prov;
}

public PopupMenuProvider getPopupMenuProvider() {
    return curPopupMenuProvider;
}

public void requestPopupMenu(PopupMenuConsumer consumer, MouseEvent e, String
context) {
    // the consumer must be a view
    if (!(consumer instanceof View))
        return;
    // this is not the stock popup so ignore it.
    if (!STOCK_CONTEXT.equals(context)) {
        super.requestPopupMenu(consumer, e, context);
        return;
    }
    curPopupMenuConsumer = consumer;
    MouseEvent newEvent = ((View) contentView).convertEventToView(this, e);
    if (curPopupMenuProvider != null)
        curPopupMenuProvider.requestPopupMenu(this, newEvent, context);
    else
        PopupMenuResources.getPopupMenuResources().requestPopupMenu(this,
newEvent, context);
}

public void customizePopupMenu(Menu menu, String context) {
    CustomLinkHandlerItem customLinkItem, customLinkItemChild;
    Enumeration items, children;
    String linkName;
    // this is not the stock popup so ignore it.
    if (!STOCK_CONTEXT.equals(context)) {
        super.customizePopupMenu(menu, context);
        return;
    }
    items = customLinkHandler.getCustomLinkItems();
    while(items.hasMoreElements()){
        customLinkItem = (CustomLinkHandlerItem) items.nextElement();
        if(customLinkItem.hasChildren()){
            Menu sm = menu.addItemWithSubmenu(customLinkItem.getName()).submenu();
            children = customLinkItem.getChildren();
            while(children.hasMoreElements()){
                customLinkItemChild = (CustomLinkHandlerItem) children.nextElement();
                linkName = customLinkItemChild.getName();
                sm.addItem(linkName, linkName, this);
            }
        }
        else {
            linkName = customLinkItem.getName();
            menu.addItem(linkName, linkName, this);
        }
    }

    if ((curPopupMenuConsumer != null) && (curPopupMenuConsumer != this))
        curPopupMenuConsumer.customizePopupMenu(menu, context);
}

```

```

ChannelWindowView.java
/**
 * TextFieldOwner interface methods, for the find panel
 */
public void selectionDidChange(Textview tv) {}

public void textDidChange(Textview tv, Range r) {}

public void textEditingDidBegin(Textview tv) {}

public void textEditingDidEnd(Textview tv) {}

public void textWillChange(Textview tv, Range r) {}

public void textEditingDidBegin(TextField tf) {}

public void textEditingDidEnd(TextField tf, int endCondition, boolean
contentsChanged) {
    if (tf == findTextField)
        if (endCondition == TextFieldOwner.RETURN_KEY) {
            if (startNewSearch)
                findButton.click();
            else
                findNextButton.click();
        }
}

public boolean textEditingWillEnd(TextField tf, int endCondition, boolean
contentsChanged) {
    return true;
}

public void textWasModified(TextField tf) {
    if (tf == findTextField) {
        if (!startNewSearch) {
            startNewSearch = true;
            contentView.resetFindSettings();
        }
    }
}

/**
public void toggleSoundButtonTo(String _stNewCommand){
    if(ChannelWindowViewMenuToolbarSet.BUTTON_COMMAND_SOUND_ON.equals(_stNewCommand)){
        latchingButtonSound.setImage(ChannelResources.BITMAP_SOUND_ON_SMALL);
latchingButtonSound.setCommand(ChannelWindowViewMenuToolbarSet.BUTTON_COMMAND_SOUND_OFF);
latchingButtonSound.setHintString(ChannelWindowViewMenuToolbarSet.BUTTON_TEXT_SOUND_ON);
    }
    else if(ChannelWindowViewMenuToolbarSet.BUTTON_COMMAND_SOUND_OFF.equals(_stNewCommand)){
        latchingButtonSound.setImage(ChannelResources.BITMAP_SOUND_OFF);
latchingButtonSound.setCommand(ChannelWindowViewMenuToolbarSet.BUTTON_COMMAND_SOUND_ON_ALERT)

```

ChannelWindowView.java

```

;
latchingButtonSound.setHintString(ChannelWindowViewMenuToolBarSet.BUTTON_TEXT_SOUND_OFF);
    }
    else if
(ChannelWindowViewMenuToolBarSet.BUTTON_COMMAND_SOUND_ON_ALERT.equals(_stNewCommand)){
        latchingButtonSound.setImage(ChannelResources.BITMAP_SOUND_ON_ALERT);
latchingButtonSound.setCommand(ChannelWindowViewMenuToolBarSet.BUTTON_COMMAND_SOUND_ON);
latchingButtonSound.setHintString(ChannelWindowViewMenuToolBarSet.BUTTON_TEXT_SOUND_ON_ALERT)
;
    }
}

    public void togglewindowButtonTo(String _stNewCommand){
        if(ChannelWindowViewMenuToolBarSet.BUTTON_COMMAND_WINDOW_ON.equals(_stNewCommand)){
            latchingButtonWindow.setImage(ChannelResources.BITMAP_WINDOW_ON_SMALL);
latchingButtonWindow.setCommand(ChannelWindowViewMenuToolBarSet.BUTTON_COMMAND_WINDOW_OFF);
latchingButtonWindow.setHintString(ChannelWindowViewMenuToolBarSet.BUTTON_TEXT_WINDOW_ON);
        }
        else if(ChannelWindowViewMenuToolBarSet.BUTTON_COMMAND_WINDOW_OFF.equals(_stNewCommand)){
            latchingButtonWindow.setImage(ChannelResources.BITMAP_WINDOW_OFF_SMALL);
latchingButtonWindow.setCommand(ChannelWindowViewMenuToolBarSet.BUTTON_COMMAND_WINDOW_ON_ALERT);
latchingButtonWindow.setHintString(ChannelWindowViewMenuToolBarSet.BUTTON_TEXT_WINDOW_OFF);
        }
        else if
(ChannelWindowViewMenuToolBarSet.BUTTON_COMMAND_WINDOW_ON_ALERT.equals(_stNewCommand)){
            latchingButtonWindow.setImage(ChannelResources.BITMAP_WINDOW_ON_ALERT_SMALL);
latchingButtonWindow.setCommand(ChannelWindowViewMenuToolBarSet.BUTTON_COMMAND_WINDOW_ON);
latchingButtonWindow.setHintString(ChannelWindowViewMenuToolBarSet.BUTTON_TEXT_WINDOW_ON_ALERT)
T);
        }
    }
}
*/

//public ChannelWindowController getController() {
//    return controller;
//}

//part of sean's custom input panel hack. -bski 2.1.99
//public void setController(ChannelWindowController aController) {
//    controller = aController;
//    if (controller != null)
//        resolveHasCustomPanel();
//    applyChannelSettings();
//}

//protected void resolveHasCustomPanel() {
//    if ((Interchange.panelPreferences != null) &&
(Interchange.panelPreferences.get(getChannelName().toUpperCase()) != null))
//        toolbarSet.setCustomInputPanelEnabled(true);
//    else
//        toolbarSet.setCustomInputPanelEnabled(false);
//}

```

FilteredChannel.java

```
package core;

/** Channel class
 *
 * Channel is a container for all data pertaining to a filtered channel;
 *
 * @auth Sean willson
 *
 * @version .1
 * @see
 */

// External Imports
import java.lang.*;
import java.util.*;
import java.io.Serializable;
import service.filter.*;

public class FilteredChannel extends Channel implements Serializable, Cloneable {
    protected FilterExpression filterExpression = null;
    static final long serialVersionUID = 20L;

    // Constructors

    public FilteredChannel() {
        this( null, null );
    }

    public FilteredChannel ( String _name ) {
        this( _name, null );
    }

    public FilteredChannel( FilteredChannel _channel ) {
        super( _channel.getName(), null, new Vector(1));
        if ( _channel.getFilterExpression() != null ) {
            filterExpression = (FilterExpression)_channel.getFilterExpression().clone();
        }
    }

    public FilteredChannel( String _name, FilterExpression _filterExpression )
    {
        super( _name, null, new Vector(1));
        this.filterExpression = _filterExpression;
    }

    // Methods

    public void copy(FilteredChannel _filteredChannel){
        this.name = _filteredChannel.getName();
        this.filterExpression = _filteredChannel.getFilterExpression();
    }

    public void copyModes(Channel _channel){}

    public Object clone() {
        return ( new FilteredChannel(this) );
    }

    // this should go away
    public Channel getChannel(Channel _channel) {
        Enumeration _members = getMembers();

        while(_members.hasMoreElements()) {
            Channel currentChannel = (Channel)_members.nextElement();

            if ( currentChannel.equals( _channel ) ) {
                return currentChannel;
            }
        }
    }
}
```

```

    }
    return null;
}

public Enumeration getMembers() {
    if(filterExpression == null)
        return null;

    Vector _vaMembers = new Vector();
    Enumeration _enumeration = this.filterExpression.getUsers();
    while(_enumeration.hasMoreElements()){
        _vaMembers.addElement(((FilterExpressionUserPart)_enumeration.nextElement()).getUser());
    }

    _enumeration = this.filterExpression.getChannels();
    while(_enumeration.hasMoreElements()){
        _vaMembers.addElement(((FilterExpressionChannelPart)_enumeration.nextElement()).getChannel());
    }

    return _vaMembers.elements();
}

public void removeAllChannels() {
    super.removeAllMembers();
}

public void clean(){
    /*
    Enumeration _enumeration = this.filterExpression.getUsers();
    while(_enumeration.hasMoreElements()){
        ((FilterExpressionUserPart)_enumeration.nextElement()).getUser().clean();
    }

    _enumeration = this.filterExpression.getChannels();
    while(_enumeration.hasMoreElements()){
        ((FilterExpressionChannelPart)_enumeration.nextElement()).getChannel().clean();
    }
    */
    setJoined(false);
}

public void clear() {
    super.clear();
}

public Enumeration getChannels() {
    if(filterExpression == null)
        return null;
    Vector _vaChannels = new Vector();
    Enumeration _enumeration = this.filterExpression.getChannels();
    while(_enumeration.hasMoreElements()){
        _vaChannels.addElement(((FilterExpressionChannelPart)_enumeration.nextElement()).getChannel());
    }

    return _vaChannels.elements();
}

```

FilteredChannel.java

```
    }

    public Vector getUsers() {
    if(filterExpression == null)
        return null;
        Vector _vaUsers = new Vector();
        Enumeration _enumeration = this.filterExpression.getUsers();
        while(_enumeration.hasMoreElements()){
        _vaUsers.addElement(((FilterExpressionUserPart)_enumeration.nextElement()).getUser());
        }
        return _vaUsers;
    }

    public void setFilterExpression(FilterExpression newFilterExpression) {
        filterExpression = newFilterExpression;
        if(filterExpression != null) {
            filterExpression.setFilteredChannelName(this.getName());
        }
    }

    public FilterExpression getFilterExpression() {
        return(filterExpression);
    }
}
```

```
package ui.controllers.channel;
```

```
import java.util.*;
import netscape.application.*;
```

```
import COM.swissbank.util.*;
import COM.swissbank.notification_center.*;
import COM.swissbank.adapter.AdapterEvent;
import COM.swissbank.adapter.Adapter;
import COM.swissbank.widgets.ifc.messagebox.MessageBox;
import COM.swissbank.widgets.ifc.text.Paragraph;
```

```
import adapter.*;
import adapter.service.*;
import listener.*;
import listener.service.*;
import ui.views.channel.*;
import ui.views.displaysettings.ChannelDisplaySettingsDialog;
import ui.views.dock.*;
import ui.views.*;
import ui.controllers.channellist.ChannelCreationController;
import core.*;
import preferences.*;
import service.database.*;
import service.filter.*;
import util.filetransfer.*;
import ui.controllers.*;
import service.util.ChannelRepository;
```

```
import util.Globals;
import util.Debug;
import preferences.ChannelSettings;
import listener.PreferencesListener;
```

```
public class FilteredChannelWindowController extends ChannelWindowController implements
Notifiable,
```

```
ContentListener,
ModifyChannelListener,
MemberListListener,
```

```
RemoveMemberListener,
```

```
ModifyMemberListener{
```

```
    private boolean bExpressionAddedToFilterEngine = false;
```

```
public FilteredChannelWindowController(core.Channel _channel) {
    super(_channel);
}
```

```
public void addContent(String mes) {
    int MAX_MESSAGE_LENGTH = 400;
    boolean isAlert = false;
    // is the message an alert
    if (mes.indexOf(util.Globals.IRC_ALERT_CHARACTER) > -1)
        isAlert = true;
    // loop through MAX_MESSAGE_LENGTH characters at a time of the message sending them out
    // in chunks. This allows long messages
    for (int i = 0; i <= (mes.length() / MAX_MESSAGE_LENGTH); i++) {
        try {
            String parsedMessage = null;
            // parse the message of the right size out of the original message
            if (mes.length() <= MAX_MESSAGE_LENGTH) {
                i++;
                parsedMessage = mes;
            }
            else {
```

FilteredChannelWindowController.java

```

if (i == 0) {
    parsedMessage = mes.substring( 0, MAX_MESSAGE_LENGTH );
}
else if (mes.length() <= ((i + 1) * MAX_MESSAGE_LENGTH)) {
    parsedMessage = mes.substring( i * MAX_MESSAGE_LENGTH, mes.length() );
}
else {
    parsedMessage = mes.substring( i * MAX_MESSAGE_LENGTH,
                                   ((i + 1) * MAX_MESSAGE_LENGTH) );
}
}
// if there is a message to send then send it
if (!"".equals(parsedMessage)) {
    // go through the member list and issue the message to each member via the backend
    Enumeration e = ((FilteredChannel)channel).getMembers();
    Member member = null;
    ContentAdapter adapter = null;
    Content content = null;
    while ((e != null) && e.hasMoreElements()) {
        // get the next element
        member = (Member)e.nextElement();
        // create an adapter for the message
        adapter = (ContentAdapter)
Recycler.getSingleton().getObject(ContentAdapter.class);
        // create a content object for the message and fill it in
        content = new Content();
        content.setAuthor(currentUser.getName());
        content.setContent(parsedMessage);
        content.setChannel(member.getName());
        if (isAlert)
            content.setAuthorType(Content.ALERT);
        else
            content.setAuthorType(Content.SELF);
        ((ContentAdapter) adapter).setContent(content);
        // send it to the individual window if it is open
        nc.enqueueNotification((new Notification(member.getName().toLowerCase(),(Object)
adapter))));
        // send it to the backend for it to go out to the network and to the filter
engine
        nc.enqueueNotification((new Notification(Globals.NC_SERVICE +
this.interchangeSettings.getNickname(),(Object) adapter))));
    }
}
}
catch (Exception e) {
    System.err.println("Exception: " + e);
    e.printStackTrace();
}
}
}

public void kill(){
    super.kill();
    try {
        // do this so that a new, un saved filter channel's filter expression will get saved
        // ((FilteredChannel)channel).clean();
        //archivewindowPosition();

FilterEngine.getSingleton().removeFilterExpression(this.channel.getName());

        // filter channel does not get a PART message, remove the user from the channel here
        // so that ChannelRepository knows that the current filtered channel is not joined
        ChannelRepository.getSingleton().removeMemberFromChannel(getCurrentUser(),
getChannel());

        adapter =
(ModifyChannelAdapter)Recycler.getSingleton().getObject(ModifyChannelAdapter.class);
        ((ModifyChannelAdapter)adapter).setChannel(this.channel);
        nc.enqueueNotification((new

```



```

        FilteredChannelWindowController.java
Notification(util.Globals.NC_COMMAND_CHANNELS, (Object)adapter));
    }
    catch (Exception _exception){
        System.err.println("FilteredChannelWindowController.kill(): " + _exception);
        _exception.printStackTrace();
    }
}

public Member getCurrentlySelectedMember(){
    return ((FilteredChannelWindowView)view).getSelectedMember();
}

/*
public void handleSpecificPrivate(Object _object) {
    //if(core.Interchange.EXTERNAL_CDT)
    // return;

    if((_object == null) || (_object instanceof Channel))
        handleNonSpecificPrivate();

        PrivateChannel _privateChannel = new
PrivateChannel(((User)_object).getName());

        try {
            this.adapter = (AddMemberAdapter)
Recycler.getSingleton().getObject(AddMemberAdapter.class);
            ((AddMemberAdapter)this.adapter).setMember(_privateChannel);
            nc.enqueueNotification((new Notification(Globals.NC_SERVICE +
this.interchangeSettings.getNickname(), (Object) adapter)));
        }
        catch (Exception e) {
            System.err.println("Unable to create adapter class! " + e.toString());
        }
    }
*/

public void handleChannelProperties() {
    ChannelCreationController.getSingleton();
    try {
        Adapter adapter =
(EditChannelAdapter)Recycler.getSingleton().getObject(EditChannelAdapter.class);
        ((EditChannelAdapter)adapter).setChannel(getChannel());
        ThreadedNotificationCenter.DefaultThreadedCenter().enqueueNotification(new
Notification(Globals.NC_COMMAND_EDIT_FILTER, adapter));
    }
    catch (Exception e){
        System.out.println("Exception (handleChannelProperties): " + e);
        e.printStackTrace();
    }
}

public void setCurrentUser(User user) {
    currentUser = user;
    this.view.setOwningUser(this.currentUser);
}

protected view getView(){
    if(this.view == null){
        this.view = new FilteredChannelWindowView();
    }
    return (view)this.view;
}

public void requestMemberList(Class _class){}

```

```

public void didReceiveMemberList(String _stUser, Enumeration _eMembers){
    while(_eMembers.hasMoreElements()){
        addMember((Member)_eMembers.nextElement());
    }
}

public void databaseTransaction(DBObject dbObject) {
    if (bwindowIsClosing)
        return;

    if ( dbObject.getObject() != null ||
        (channel != null && ((FilteredChannel)channel).getFilterExpression() != null) )
        super.databaseTransaction(dbObject);
    else {
        // if there's no filter expression
        // set the filter expression of the FilteredChannel to be the default filter
        // for the filtered channel
        synchronized (closeLock) {
            Debug.report(this.getClass(), Debug.INFO_DEBUG, "channel: "+channel.getName()+"
Filter expression is corrupted, replace with default filter");
            if (getChannelSettings() == null) {
                // create a default channel settings
                setChannelSettings(new ChannelSettings());
                getChannelSettings().setName(this.channel.getName());
            }
            // create a default filter
            if (getChannelSettings().getObject() == null) {
                // if the database filter setting is corrupted, we will create a default filter to
replace it
                FilteredChannel channel = new FilteredChannel(this.channel.getName(), null);
                getChannelSettings().setObject(channel);

                /*
                // post a message to the server saying that the filter has been replaced
                Content content = new Content();
                content.setAuthorType(Content.INTERCHANGE);
                content.setAuthor("*****");
                content.setContent("Your filter is corrupted for this channel, Interchange has
replaced it with a default filter");
                addContent(content);
                */
            }

            getChannelSettings().setShowChannel(new Boolean(true));
            applyChannelSettings();
            hasReceivedPrefs = true;

            if(this.view instanceof PrivateChannelWindowView){
                while(!queueContent.isEmpty()){
                    addContent((Content)queueContent.pop());
                }
            }
            else{
                // process the queue'd up messages that came in while waiting
                while(!queueContent.isEmpty()){
                    this.view.appendContent((Content)queueContent.pop());
                }
                // applyChannelSettings();
            }
        }
    }
}
}
}

```

```

protected void applyChannelSettings(){
    Debug.report(this.getClass(), Debug.INFO_DEBUG, "apply channel settings");
}

```

```

        FilteredChannelwindowController.java
this.getChannelSettings().setType(Globals.TYPE_FILTERED);

/*
 * The second part of this if is the important bit. If the
 * channel object, passed in at instantiation, is not filled in,
 * we have to take the settings from the database. The only way
 * to achieve this is double-clicking on a filtered channel in the
 * channel list when it hasn't been joined. Other cases mean that we
 * have constructed the filter or edited it.
 */

    Debug.report(this.getClass(), Debug.INFO_DEBUG, "added filter " +
this.bExpressionAddedToFilterEngine);
    Debug.report(this.getClass(), Debug.INFO_DEBUG, "this.getChannelSettings().getObject() =
" + this.getChannelSettings().getObject());
    Debug.report(this.getClass(), Debug.INFO_DEBUG,
"((FilteredChannel)this.channel).getFilterExpression() = " +
((FilteredChannel)this.channel).getFilterExpression());

    if((this.getChannelSettings().getObject() != null)) { //&&
(((FilteredChannel)this.channel).getFilterExpression() == null)){
        Debug.report(this.getClass(), Debug.INFO_DEBUG, "setting new channel object");
        this.channel =
(FilteredChannel)this.getChannelSettings().getObject();
        if (((FilteredChannel)channel).getFilterExpression() == null) {
            ((FilteredChannel)channel).setFilterExpression(new FilterExpression
(this.channel.getName(), false));

            // post a message to the server saying that the filter has been replaced
            Content content = new Content();
            content.setAuthorType(Content.INTERCHANGE);
            content.setAuthor("*****");
            content.setContent("Your filter is corrupted for this channel, Interchange has
replaced it with a default filter");
            addContent(content);
            getChannelSettings().setObject(channel);
        }
    }

    if(this.bExpressionAddedToFilterEngine) {
        Debug.report(this.getClass(), Debug.INFO_DEBUG, "removing old filter expression");
        FilterEngine.getSingleton().removeFilterExpression(this.channel.getName());
        this.bExpressionAddedToFilterEngine = false;
    }

    Debug.report(this.getClass(), Debug.INFO_DEBUG, " added filter " +
this.bExpressionAddedToFilterEngine);
    Debug.report(this.getClass(), Debug.INFO_DEBUG, " this.getChannelSettings().getObject() =
" + this.getChannelSettings().getObject());
    Debug.report(this.getClass(), Debug.INFO_DEBUG, "
((FilteredChannel)this.channel).getFilterExpression() = " +
((FilteredChannel)this.channel).getFilterExpression());

    if(!this.bExpressionAddedToFilterEngine) &&
(((FilteredChannel)this.channel).getFilterExpression() != null)){
        Debug.report(this.getClass(), Debug.INFO_DEBUG, "adding filter expression");
        //modified by bski 1.11.99
        FilterEngine.getSingleton().addFilterExpression(
this.channel.getName(),
(FilterExpression)((FilteredChannel)this.channel).getFilterExpression());//.clone()
);
        this.bExpressionAddedToFilterEngine = true;
    }

    Enumeration e = ((FilteredChannel)this.channel).getMembers();

    if(e != null)
        ((FilteredChannelwindowView)this.view).addMembers(e);

    if(this.getChannelSettings().getIsDocked().booleanValue()){

```

FilteredChannelWindowController.java
 TargetFacility.sendCommand(util.Globals.COMMAND_SDOCK_DOCK_CHANNEL,

```
this);
    } else {
        positionWindow();
        bringToFront();
        //getWindow().show();
    }
    view.setDisplayable(true);
    view.resetSizes();
}

public void modifyChannel(Channel channel){
    ((FilteredChannelWindowView)view).updateUserList(channel);
}

public void addMember(Member _member) {
    ((FilteredChannelWindowView)this.view).addMember(_member);
}

    public void removeMember(Member _aMember){
        ((FilteredChannelWindowView)this.view).removeMember(_aMember);
    }

public void modifyMember(Member _member){
    ((FilteredChannelWindowView)this.view).modifyMember(_member);
}

public void clearUserList() {
    ((FilteredChannelWindowView)view).clearUserList();
}

/**
settings    *      Returns a ChannelSettings object with the current settings. Most
            *      are straight from the database settings, but some are from the view.
            */
    protected void updateChannelSettings() {
        super.updateChannelSettings();
        this.getChannelSettings().setObject(this.channel);
    }
}
```

```

package ui.views.channel;

import ui.controllers.*;
import ui.controllers.channel.*;
import netscape.application.*;
import netscape.util.Vector;
import java.util.Enumeration;

import COM.swissbank.adapter.*;
import COM.swissbank.widgets.ifc.core.*;
import COM.swissbank.widgets.ifc.text.*;
import COM.swissbank.widgets.ifc.toolbar.*;
import COM.swissbank.widgets.ifc.buttons.*;
import COM.swissbank.widgets.ifc.splitter.*;
import COM.swissbank.util.TargetFacility;
import COM.swissbank.util.*;
import COM.swissbank.notification_center.*;

import util.Globals;
import adapter.*;
import ui.views.channellist.member.*;
import core.*;
//import ui.views.dock.Dockable;
import preferences.*;
import ui.views.ChannelResources;
import ui.controllers.channellist.ChannelCreationController;

/**
 * @(#)FilteredChannelWindowView.java
 *
 * FilteredChannelWindowView is the main view use to control and display filtered
 * channels. The user list contains members.
 *
 * Copyright 1998 Warburg Dillon Read. All Rights Reserved.
 *
 * @author D. Zaret, Don Bora, Sean Willson
 */
public class FilteredChannelWindowView extends ChannelWindowView implements
MemberList.MemberListOwner {

    private static final int MEMBERLIST_WIDTH = 85;
    protected static final String FILTERED_CHANNEL_ICON = "filtered_channel_small.gif";
    private MemberList memberList;

    public void applyChannelSettings() {
        super.applyChannelSettings();
        setShowJoinPartEnabled(false);
        toolbarSet.setChannelPropertiesEnabled(true);
    }

    // allow user list
    public boolean getShowUserListEnabled() {
        return true;
    }

    public boolean hasMemberList(){
        return true;
    }

    public void clearUserList() {
        this.memberList.removeAllItems();
    }

    public String getSelectedItem() {

```

```

        FilteredChannelWindowView.java
    } return this.memberList.getCurrentlySelectedMember().getName();
}

public void updateUserList(Channel channel) {
    memberList.removeAllItems();
    Enumeration e = channel.getMembers();
    addMembers(e);
}

public boolean getKickEnabled() {
    return false;
}

public boolean getMakeOperatorEnabled() {
    return false;
}

public boolean getChannelPropertiesEnabled() {
    return true;
}

//public void updateUserMenu() {
/*try {
    String selectedUser = "";
    if (this.memberList.getCurrentlySelectedMember() != null)
        selectedUser = this.memberList.getCurrentlySelectedMember().getName();
    toolbarSet.updateUserMenu(selectedUser, ((User)
this.memberList.getCurrentlySelectedMember()).isPrivileged());
}
catch (Exception e) {
    System.out.println("ChannelWindowView.updateUserMenu() died");
}
} */
//}

//public void updateChannelMenu() {
// toolbarSet.updateChannelMenu(getChannelName());
//}

public Bitmap getChannelBitmap(){
    return webUtilities.findBitmap(FILTERED_CHANNEL_ICON);
}

public void addMember(Member _member) {
    try {
        this.memberList.addItem(_member);
    } catch (InvalidMemberException e){
        System.err.println("Exception: " + e);
        e.printStackTrace();
    }
}

public void addMembers(Enumeration enum) {
    if (enum == null)
        return;
    try {
        this.memberList.addItems(enum);
    } catch (InvalidMemberException e){
        System.err.println("Exception: " + e);
        e.printStackTrace();
    }
}
}

```

FilteredChannelWindowView.java

```
/**
 * Overridden from CustomView. Called from the constructor when the widgets
 * need to be created. Note how here they are created and configured, but not
 * positioned or sized.
 */
protected void createWidgets() {
    super.createWidgets();
    this.memberList = new MemberList();
    this.memberList.setOwner(this);
    this.memberList.setPopupMenuProvider(this);
    this.memberList.configForChannelWindow();
    container.addView(this.memberList);
}

protected int getUserListWidth() {
    return MEMBERLIST_WIDTH;
};

// ignore for filtered
public boolean getShowJoinPart() {
    return false;
}

/**
 * Overridden from CustomView. Called when this view needs to be resized.
 * All widget layout and sizing should be performed in this method.
 */
public void resetSizes(){
    super.resetSizes();
    if (!displayable) return;
    if (getShowUserList()) {
        if (getShowCustomPanel() && (getCustomInputPanel() != null))
            memberList.setSize(MEMBERLIST_WIDTH, toHeight - menuToolBarHeight - SMALL_MARGIN -
customInputPanel.PREFERRED_HEIGHT);
        else
            memberList.setSize(MEMBERLIST_WIDTH, toHeight - menuToolBarHeight - SMALL_MARGIN);
        memberList.moveTo(totWidth - memberList.width() - SMALL_MARGIN, splitView.y());
    }
    else { // hide the user list
        memberList.setSize(0,0);
        memberList.moveTo(totWidth - SMALL_MARGIN,-1); // move it to the right edge so the
contentView will size itself correctly
    }
    // set the view dirty since it may need to be repainted
    container.setDirty(true);
    //updateUserMenu();
}

public void modifyMember(Member _member){
    try {
        this.memberList.memberChanged(_member);
    }
    catch (Exception e) {
        System.out.println("Exception " + e);
        e.printStackTrace();
    }
}

//public void setOwningUser(User user){
//    super.setOwningUser(user);
//    if (user == null)
//        return;
//    toolbarSet.setMakeOperatorEnabled(false);
//}
```

```

                                FilteredChannelWindowView.java
// toolbarSet.setKickEnabled(false);
// toolbarSet.setChannelPropertiesEnabled(true);
//}

public boolean removeMember(Member _member ) {
    boolean _bReturnValue = true;
    try {
        this.memberList.removeItem(_member);
    } catch (InvalidMemberException e){
        System.out.println("Exception: " + e);
        e.printStackTrace();
        _bReturnValue = false;
    }
    resetSizes();
    return _bReturnValue;
}

public Member getSelectedMember(){
    Member _member = null;
    if (getShowUserList())
        _member = this.memberList.getCurrentlySelectedMember();
    return _member;
}

public netscape.util Enumeration getInvisibleChannels() {
    return null;
}

public void userListDidSelectInvite(User user, Channel channel) {}

public void memberListKeySearch(MemberList list, String searchString, int status) {}

    public void memberListConfigMenu(MemberList list) {}

    public void memberListSelectedMember(MemberList list, Member member) {
    if (member instanceof User)
        if (getOwner() != null)
            ((ChannelWindowViewOwner) getOwner()).handleSpecificPrivate((User) member);
        //(((FilteredChannelWindowController) getController()).handleSpecificPrivate(member);
    }

    public void memberListSelectedMemberByReturn(MemberList list, Member member){
        if (member instanceof User)
            if (getOwner() != null)
                ((ChannelWindowViewOwner) getOwner()).handleSpecificPrivate((User) member);
            //(((FilteredChannelWindowController)
getController()).handleSpecificPrivate(member);
    }

    public void memberListSelectedMemberByDbClick(MemberList list, Member member){
        if (member instanceof User)
            if (getOwner() != null)
                ((ChannelWindowViewOwner) getOwner()).handleSpecificPrivate((User) member);
            //(((FilteredChannelWindowController)
getController()).handleSpecificPrivate(member);
    }

//public void performCommand(String stCommand, Object object) {
//    if (ChannelWindowViewMenuToolBarSet.MENU_EDIT_CHANNEL.equals(stCommand))
//        handleChannelProperties();

```



```

// else
//     super.performCommand(stCommand,object);
//}

/*
protected void handleChannelProperties() {
    try {
        ChannelCreationController.getSingleton();
        Adapter adapter =
(EditChannelAdapter)Recycler.getSingleton().getObject(EditChannelAdapter.class);
        ((EditChannelAdapter)adapter).setChannel(getController().getChannel());
        ThreadedNotificationCenter.DefaultThreadedCenter().enqueueNotification(new
Notification(Globals.NC_COMMAND_EDIT_FILTER, adapter));
    }
    catch (Exception e){
        System.out.println("Exception (handleChannelProperties): " + e);
        e.printStackTrace();
    }
}
*/

//public void handleNonSpecificPrivate(){
//    //     if(getOwner() != null)
//    //         ((ChannelWindowViewOwner) getOwner()).handleNonSpecificPrivate();
//}

// from the MemberListOwner interface
public void handleSpecificPrivate(User user) {
    if (getOwner() != null)
        ((ChannelWindowViewOwner) getOwner()).handleSpecificPrivate(user);
}

// from the MemberListOwner interface
public void handleSpecificWhois(User user) {
    if (getOwner() != null)
        ((ChannelWindowViewOwner) getOwner()).handleSpecificWhois(user);
}

// these aren't accessible from a Filtered channel
public void handleMakeOperator() {}
public void handleKick() {}
public void handleInvite(Channel channel) {}
}

```

```

package ui.views.channel.custom;

import netscape.application.*;
import java.awt.Toolkit;
import java.net.*;
import java.io.*;

import core.Interchange;

import COM.swissbank.widgets.ifc.core.*;
import COM.swissbank.widgets.ifc.text.*;
import COM.swissbank.widgets.ifc.toolbar.*;
import COM.swissbank.widgets.ifc.buttons.*;
import COM.swissbank.widgets.ifc.splitter.*;
import COM.swissbank.util.TargetFacility;

public class HTTPInputPanel extends CustomInputPanel
{
    public static final boolean DEBUG = false;

    public HTTPInputPanel(){
        super();
    }

    /**
     * This method is used to return the result of going to a webpage. This webpage
     * can be authenticated as well as one that requires some input.
     *
     * @param login The login id required to goto this webpage. If null then assumes that
     * the page is not authenticated.
     * @param password The associated password for the login.
     * @param destinationURL The URL that you want to see or post to.
     * @param data The data that you wish to Post to the destination URL.
     * @return The content of the resulting page.
     */
    public String handleHTTPPost(String login, String password, String destinationURL, String
data) {
        // Perform the HTTP Post and retrieve the results
        String response="";
        try
        {
            URL postURL = new URL(destinationURL);
            URLConnection connect = postURL.openConnection();
            // set it up to read input and output
            connect.setDoInput(true);
            connect.setDoOutput(true);
            // turn on user interaction
            connect.setAllowUserInteraction(true);
            // turn caching off
            connect.setUseCaches(false);
            // do we care about a password
            if ((login != null) && (password != null)) {
                // Encode Password String
                String userPassword = login + ":" + password;
                String encoding = new sun.misc.BASE64Encoder().encode(userPassword.getBytes());
                // Need to work with URLConnection to set request properties
                connect.setRequestProperty("Authorization", "Basic " + encoding);
            }
            // do we care about sending any data
            if (data != null) {
                // set the type of content being sent
                connect.setRequestProperty("Content-type", "application/octet-stream");
                // set the length of the content
                connect.setRequestProperty("Content-length", Integer.toString(data.length()));
                // get the output stream and send it all to the server
                DataOutputStream outputStream = new DataOutputStream(connect.getOutputStream());
                outputStream.write(data.getBytes());
                outputStream.flush();
            }
        }
        catch (Exception e) {
            // Handle exception
        }
        return response;
    }
}

```

```

        outStream.close();
    }
    // read the return data from the server
    DataInputStream inStream = new DataInputStream(connect.getInputStream());
    // int response_length = connect.getContentLength();
    // byte[] bresponse = new byte[response_length];
    // inStream.readFully(bresponse,0,response_length);
    // response = new String(bresponse);
    StringBuffer responseStringBuf = new StringBuffer("");
    String responseLine;
    while ( (responseLine = inStream.readLine()) != null) {
        responseStringBuf.append(responseLine);
        responseStringBuf.append("\n");
    }
    response = responseStringBuf.toString();
    inStream.close();
    if (DEBUG) {
        System.out.println("Result after POST" + "\n " + response);
    }
}
catch (Exception e) {
    System.out.println("Error Posting: " + e);
    e.printStackTrace();
}

    return response;
}
}

```

```

CustomInputPanel.java
package ui.views.channel.custom;

import netscape.application.*;
import COM.swissbank.widgets.ifc.core.*;

public class CustomInputPanel extends CustomView
{
    protected ContainerView    panelContainer;
    protected CustomInputPanelOwner owner;
    public int PREFERRED_HEIGHT = 118;

    public static interface CustomInputPanelOwner extends Owner {
        public void messageEntered(String msg);
        public void addMessage(String msg);
        public String getChannelName();
    }

    public CustomInputPanel() {
        super();
    }

    public void setOwner(CustomInputPanelOwner anOwner) {
        this.owner = anOwner;
    }

    public Owner getOwner() {
        return this.owner;
    }

    protected void createwidgets() {
        // create the panel view
        panelContainer = new ContainerView();
        panelContainer.setBorder(new BezelBorder(BezelBorder.LOWERED));
        // add this container view to the main view
        addSubview(panelContainer);
    }

    public void resetPanelSize(int width, int height) {
    }

    public void resetsizes(){
        panelContainer.sizeTo(totwidth, totHeight);
        panelContainer.moveTo(0,0);
        resetPanelSize(totwidth, totHeight);
    }

    public void setupFocus() {
    }

    public void performCommand(String stCommand, Object object) {
    }
}

```

```

package ui.views.channel.custom;

import com.oroinc.text.regex.*;
import com.ibm.xml.parser.*;
import COM.swissbank.util.webutilities;
import service.util.*;
import netscape.util.*;
import java.io.*;
import core.Interchange;
import ui.views.channel.custom.CustomLinkHandlerItem;
import preferences.xml.XMLPreferenceReader;
import service.startup.*;

public class CustomLinkHandler
{
    private static CustomLinkHandler customLinkHandler;
    protected Hashtable linkHash, spHash, oSuffixHash, nSuffixHash;
    protected String stockString = null;
    protected String replacementString = null;
    protected CustomLinkHandlerItem customLinkItems = null;
    public static final String CUSTOM_LINKS_FILENAME = "symbollinks.xml";
    public static final String LINK_ELEMENT = "link";
    public static final String LINKNAME_ELEMENT = "linkname";
    public static final String LINKURL_ELEMENT = "linkurl";
    public static final String PARENT_ATTRIBUTE = "parent";
    public static final String ID_ATTRIBUTE = "id";
    public static final String ROOT = "root";
    public static final String KALEIDOSCOPE = "kaleidoscope";
    public static final String REPLACEMENT = "replacementstring";
    public static final String SPX_PREFIX = "spxprefix";
    public static final String O_SUFFIX = "osuffix";
    public static final String N_SUFFIX = "nsuffix";

    private CustomLinkHandler() {
        this(null);
    }

    private CustomLinkHandler(File file) {
        linkHash = new Hashtable(37);
        spHash = new Hashtable(11);
        oSuffixHash = new Hashtable(11);
        nSuffixHash = new Hashtable(11);

        customLinkItems = new CustomLinkHandlerItem(ROOT,true);
        if(file == null)
            loadFile(getDefaultFile());
        else
            loadFile(file);
    }

    public static CustomLinkHandler getSingleton() {
        if(customLinkHandler == null)
            customLinkHandler = new CustomLinkHandler();
        return customLinkHandler;
    }

    protected File getDefaultFile(){
        File file;
        if (ICRunTimeContext.getMarimbaContext() != null){
            file = new
File(ICRunTimeContext.getMarimbaContext().getDataDirectory() + ((Interchange)
netscape.application.Application.application()).getLogin(), CUSTOM_LINKS_FILENAME);
        }
        else if (ICRunTimeContext.runTimeIsSpawnedJRE()) {
            file = new File(System.getProperty("marimba.channelDataDir") + ((Interchange)
netscape.application.Application.application()).getLogin(), CUSTOM_LINKS_FILENAME);
        }
        else{
            file = new File(/*((Interchange)
netscape.application.Application.application()).getLogin(),*/ CUSTOM_LINKS_FILENAME);
        }
    }
}

```

```

    }
    return file;
}

protected void loadFile(File file){
    XMLPreferenceReader xmlReader = null;
    TXElement linkArr[], spPrefix, oSuffix, nSuffix;
    String parent, linkName, linkURL, id;
    CustomLinkHandlerItem item;

    try {
        xmlReader = new XMLPreferenceReader(file.getPath());
    } catch (FileNotFoundException e) {
        e.printStackTrace();
        return;
    }

    replacementString = xmlReader.getDocumentElement(REPLACEMENT).getText();
    linkArr = xmlReader.getElementsNamed(LINK_ELEMENT);
    for(int i=0; i< linkArr.length; i++){
        linkName = linkArr[i].getElementNamed(LINKNAME_ELEMENT).getText();
        linkURL = linkArr[i].getElementNamed(LINKURL_ELEMENT).getText();
        spPrefix = linkArr[i].getElementNamed(SPX_PREFIX);
        oSuffix = linkArr[i].getElementNamed(O_SUFFIX);
        nSuffix = linkArr[i].getElementNamed(N_SUFFIX);

        if(spPrefix != null)
            spHash.put(linkName, spPrefix.getText());

        if(oSuffix != null)
            oSuffixHash.put(linkName, oSuffix.getText());

        if(nSuffix != null)
            nSuffixHash.put(linkName, nSuffix.getText());

        if((linkURL != null) && (linkName != null))
            linkHash.put(linkName, linkURL);
        parent = linkArr[i].getAttribute(PARENT_ATTRIBUTE);

        if(parent == null)
            return;

        if(ROOT.equals(parent)){
            id = linkArr[i].getAttribute(ID_ATTRIBUTE);
            if(id != null)
                customLinkItems.addChild(id);
        } else {
            item = customLinkItems.getChildForParentID(parent);
            if(item != null)
                item.addChild(linkName);
        }
    }
}

public boolean isCustomCommand(String command){
    return (linkHash.get(command) != null);
}

public void handleLink(String linkName, String symbol){
    String url = getUrlForLinkName(linkName, symbol);
    System.out.println("Opening url "+url);

    if(url == null)
        return;

    try {
        webUtilities.showDocument(url);
    }
    catch (Exception e){
        e.printStackTrace();
    }
}

```

```

    }
}

public Enumeration getCustomLinkItems(){
    return customLinkItems.getChildren();
}

public String getUrlForLinkName(String linkName, String symbol){
    if((symbol == null) || (linkName == null))
        return null;

    String url = (String)linkHash.get(linkName);
    symbol = modifySymbolForLinkName(symbol, linkName);
    String symbolUrl = ICPatternMatcher.getSingleton().substitute(replacementString,

        url, symbol.toUpperCase());
    return symbolUrl;
}

public void handleKScopeCall(String symbol){
    handleLink(KALEIDOSCOPE, symbol);
}

protected String modifySymbolForLinkName(String symbol, String linkName){
    String fix;
    int index;

    if((symbol == null) || (linkName == null))
        return symbol;

    if ( (index = symbol.indexOf(".")) > -1 )
        symbol = symbol.substring(0, index);

    if(symbol.toLowerCase().indexOf("SPX".toLowerCase()) > -1){
        if ((symbol.charAt(0) == '.') || (symbol.charAt(0) == '^'))
            symbol = symbol.substring(1);

        fix = (String)sphash.get(linkName);
        if(fix != null)
            return fix+symbol;
        else
            return symbol;
    } else {
        if(symbol.length() == 3)
            fix = (String)oSuffixHash.get(linkName);
        else
            fix = (String)nSuffixHash.get(linkName);

        if(fix != null)
            return symbol+fix;
        else
            return symbol;
    }
}
}

```

```

package ui.views.channel.custom;

import com.oroinc.text.regex.*;
import com.ibm.xml.parser.*;
import COM.swissbank.util.Webutilities;
import service.util.*;
import netscape.util.*;

public class CustomLinkHandlerItem implements netscape.util.Comparable
{
    protected Vector children = null;
    protected String name;
    protected boolean hasChildren = false;

    public CustomLinkHandlerItem(){
        this(null,false);
    }

    public CustomLinkHandlerItem(String name) {
        this(name,false);
    }

    public CustomLinkHandlerItem(String name, boolean isParent) {
        hasChildren = isParent;
        children = null;
        setName(name);
    }

    public int compareTo(Object linkItem){
        if(name == null)
            return 1;
        else if(linkItem == null)
            return -1;
        else if(linkItem.toString() == null)
            return -1;
        else
            return name.compareTo(((CustomLinkHandlerItem)linkItem).getName());
    }

    public String getName(){
        return name;
    }

    public String toString() {
        return getName();
    }

    public void setName(String name){
        this.name = name;
    }

    public void addChild(String name){
        if(name != null) {
            addChild(new CustomLinkHandlerItem(name));
        }
    }

    public void addParent(String id, String name){
        if((name != null) && (id != null)){
            CustomLinkHandlerItem item = new CustomLinkHandlerItem(id);
            item.setName(name);
            addChild(item);
        }
    }

    public void addChild(CustomLinkHandlerItem item) {
        if(item == null)
            return;
        if(children == null){
            children = new Vector();

```


CustomLinkHandlerItem.java

```
    }
    hasChildren = true;
    children.addElement(item);
}

public Enumeration getChildren() {
    return children.elements();
}

public boolean hasChildren() {
    return hasChildren;
}

public CustomLinkHandlerItem getChildForParentID(String id){
    CustomLinkHandlerItem item;
    for(int i=0; i < children.size(); i++){
        item = (CustomLinkHandlerItem)children.elementAt(i);
        if(item.getName().equals(id))
            return item;
    }
    return null;
}
}
```

```
package service.startup;
```

```
import java.util.*;
import java.io.*;
import java.net.*;
```

```
import COM.swissbank.util.*;
```

```
import marimba.io.*;
import util.*;
import core.*;
```

```
public class CustomPanelHandler {
```

```
    public static String customPanelFile = "custom_panel.txt";
```

```
    public CustomPanelHandler() {
```

```
        public static void loadCustomPanelPreferences() {
```

```
            String panelPreference;
```

```
            Debug.report(CustomPanelHandler.class, Debug.INFO_DEBUG, "Loading Custom Input Panel Preferences");
```

```
            boolean fileExists = false;
            int index;
```

```
            if (ICRunTimeContext.runTimeIsSpawnedJRE()) {
                File file = new File(System.getProperty("marimba.channel.dir") + customPanelFile);
                fileExists = file.exists();
            }
```

```
            else if (ICRunTimeContext.runTimeIsMarimba()) {
                File file = new File(ICRunTimeContext.getMarimbaContext().getDataDirectory(),
customPanelFile);
                fileExists = file.exists();
            }
```

```
            else {
                File file = new File("C:/Projects/Interchange 2.0/Development/app/Interchange/JAVA/"
+ customPanelFile);
                fileExists = file.exists();
            }
```

```
                if (fileExists) {
                    try {
                        URL base = null;
                        if (ICRunTimeContext.runTimeIsSpawnedJRE()) {
                            base = new URL(WebUtilities.getStandAlonePath() + customPanelFile);
                        }
```

```
                        else if (ICRunTimeContext.runTimeIsMarimba()) {
                            // base = new URL(ICRunTimeContext.getMarimbaContext().getBaseURL(),
customPanelFile);
                            base = new URL("file:/" + ICRunTimeContext.getMarimbaContext().getDataDirectory() +
customPanelFile);
                        }
```

```
                        else {
                            base = new URL("file:/C:/Projects/Interchange
2.0/Development/app/Interchange/JAVA/" + customPanelFile);
                        }
```

```
                        FastInputStream fs = new FastInputStream(base.openStream());
                        Interchange.panelPreferences = new Hashtable();
```

```
                        while ((panelPreference = fs.readLine()) != null) {
                            Debug.report(CustomPanelHandler.class, Debug.INFO_DEBUG, "Read Line: " +
panelPreference);
                        }
```

```
                        // format should be "channel objectname"
                        index = panelPreference.indexOf(" ");
                        if (index != -1) {
```

```
                            Interchange.panelPreferences.put(panelPreference.substring(0, index).toUpperCase(),
```

```

CustomPanelHandler.java
panelPreference.substring(index+1));
    }
    fs.close();
}
catch (Exception e) {
    System.out.println("Exception: " + e);
    e.printStackTrace();
}
    } else {
        System.err.println("Interchange.java: Couldn't loading CustomPanelPreferences");
    }
}
}

```

```

package ui.views.channel.custom;

import netscape.application.*;
import java.awt.Toolkit;
import java.net.*;
import java.io.*;
import java.util.*;
import com.ibm.xml.parser.*;
import preferences.xml.*;

import core.Interchange;

import COM.swissbank.widgets.ifc.core.*;
import COM.swissbank.widgets.ifc.text.*;
import COM.swissbank.widgets.ifc.toolbar.*;
import COM.swissbank.widgets.ifc.buttons.*;
import COM.swissbank.widgets.ifc.splitter.*;
import COM.swissbank.util.TargetFacility;
import COM.swissbank.util.WebUtilities;
import util.Debug;

public class CustomQInputPanel extends HTTPInputPanel implements Target,
                                                                    TextFieldOwner
{
    public static final boolean DEBUG = false;

    private String POST_URL_ADDRESS;
    private String HELP_URL_ADDRESS;
    private String DATA_URL_ADDRESS;
    private boolean dataLoaded = false;

    public static final String STATUS_HEADER = "transfer-status";
    public static final String ERROR_HEADER = "error";
    public static final String POSTED_NAME_HEADER = "postedName";

    protected final static String HEAD1 = "CustomerQ Input Form - For Use by Americas and
London";
    protected final static String HEAD2 = "For High Priority Issues, Call your local TSC";
    protected final static String SUBMIT = "Submit";
    protected final static String CLEAR = "Clear";
    protected final static String HELP = "Help";
    protected final static String TYPE = "Type";
    protected final static String SOURCE = "Source";
    protected final static String PRIORITY = "Priority";
    protected final static String SCOPE = "Scope";
    protected final static String CATEGORY = "Category";
    protected final static String CONTACT_PHONE = "Contact Phone Number";
    protected final static String CENTER = "Technical Support Center";
    protected final static String DESCRIPTION = "Description";
    protected final static String PROBLEM = "Problem";
    protected final static String PICK_ONE = "Pick One";
    protected final static String HIGH = "High";
    protected final static String HIGH_SYMBOL = "(H)";

    /*
private final static String categoryDisplayItems[] = { PICK_ONE,
    "Application", "Email", "Hardware",
    "Infrastructure", "Market Data Service",
    "Software", "Voice" };
private final static int categoryKeys[] = {0, 463, 462, 465, 466,
    467, 464, 468};

private final static String centerDisplayItems[] = { PICK_ONE,
"Chicago Trading and Corporate TSC",
    "London Trading TSC",
    "London Corporate TSC",

```

```

CustomQInputPanel.java
    "Stamford Trading TSC",
    "Stamford/New York Corporate TSC");
private final static int centerKeys[] = {0, 221, 1851, 981, 382, 881};
private final static String typeDisplayItems[] = {PICK_ONE, "Problem", "Service Request"};
private final static int typeKeys[] = {-1, 0, 1};
private final static String scopeDisplayItems[] = {PICK_ONE, "Individual",
    "Business Unit", "Regional", "Global"};
private final static int scopeKeys[] = {-1, 0, 1, 2, 3};
*/
private Vector categoryKeys;
private Vector centerKeys;
private Vector typeKeys;
private Vector scopeKeys;

protected Label
protected Label
protected Label
protected CutCopyPasteTextField
protected Label
protected Popup
protected Label
protected Popup
// protected Label
// protected Popup
// protected Label
// protected Popup
protected Label
protected Popup
protected Label
protected Label
protected Label
protected Label
protected Label
protected Label
protected CutCopyPasteTextField
protected Button
protected Button
protected Button          clearButton;

head1Label;
head2Label;
contactPhoneLabel;
contactPhoneTextField;
centerLabel;
centerPopup;
typeLabel;
typePopup;
sourceLabel;
sourcePopup;
priorityLabel;
priorityPopup;
scopeLabel;
scopePopup;
categoryLabel;
categoryPopup;
descriptionLabel;
descriptionTextField;
helpButton;
submitButton;

public CustomQInputPanel(){
    super();
    PREFERRED_HEIGHT = 220;
}

protected void createwidgets() {
    super.createwidgets();
    //create headers
    head1Label = new Label();
    head1Label.setTitle(HEAD1);
    head1Label.setJustification(Graphics.CENTERED);
    panelContainer.addSubview(head1Label);
    head2Label = new Label();
    head2Label.setTitle(HEAD2);
    head2Label.setJustification(Graphics.CENTERED);
    panelContainer.addSubview(head2Label);
    //create contact phone stuff
    contactPhoneLabel = new Label();
    contactPhoneLabel.setTitle(CONTACT_PHONE);
    contactPhoneLabel.setJustification(Graphics.LEFT_JUSTIFIED);
    panelContainer.addSubview(contactPhoneLabel);
    contactPhoneTextField = new CutCopyPasteTextField();
    contactPhoneTextField.setOwner(this);
    contactPhoneTextField.setPopupMenuProvider(this);
    panelContainer.addSubview(contactPhoneTextField);
    //create technical support center stuff
    centerLabel = new Label();
    centerLabel.setTitle(CENTER);
    centerLabel.setJustification(Graphics.LEFT_JUSTIFIED);
    panelContainer.addSubview(centerLabel);
}

```

```

centerPopup = new Popup();
/*
for (int i=0; i<centerDisplayItems.length; i++) {
    centerPopup.addItem(centerDisplayItems[i], null);
}
*/
panelContainer.addSubview(centerPopup);
// create the type stuff
typeLabel = new Label();
typeLabel.setTitle(TYPE);
typeLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(typeLabel);
typePopup = new Popup();
/*
for (int i=0; i<typeDisplayItems.length; i++) {
    typePopup.addItem(typeDisplayItems[i], null);
}
*/
panelContainer.addSubview(typePopup);

// create the scope stuff
scopeLabel = new Label();
scopeLabel.setTitle(SCOPE);
scopeLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(scopeLabel);
scopePopup = new Popup();
/*
for (int i=0; i< scopeDisplayItems.length; i++) {
    scopePopup.addItem(scopeDisplayItems[i], null);
}
*/
panelContainer.addSubview(scopePopup);

// create the category stuff
categoryLabel = new Label();
categoryLabel.setTitle(CATEGORY);
categoryLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(categoryLabel);
categoryPopup = new Popup();
/*
for (int i=0; i< categoryDisplayItems.length; i++) {
    categoryPopup.addItem(categoryDisplayItems[i], null);
}
*/
panelContainer.addSubview(categoryPopup);
// create the problem stuff
descriptionLabel = new Label();
descriptionLabel.setTitle(DESCRIPTION);
descriptionLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(descriptionLabel);
descriptionTextField = new CutCopyPasteTextField();
descriptionTextField.setOwner(this);
descriptionTextField.setPopupMenuProvider(this);
panelContainer.addSubview(descriptionTextField);
// create the help button
helpButton = new Button();
helpButton.setTitle(HELP);
helpButton.setCommand(HELP);
helpButton.setTarget(this);
panelContainer.addSubview(helpButton);
// create the submit button
submitButton = new Button();
submitButton.setTitle(SUBMIT);
submitButton.setCommand(SUBMIT);
submitButton.setTarget(this);
panelContainer.addSubview(submitButton);
// create the clear button
clearButton = new Button();
clearButton.setTitle(CLEAR);
clearButton.setCommand(CLEAR);

```

```

clearButton.setTarget(this);
panelContainer.addSubview(clearButton);
// setup the forward tabbing
descriptionTextField.setTabField(contactPhoneTextField);
contactPhoneTextField.setTabField(descriptionTextField);
// setup the back tabbing
contactPhoneTextField.setBacktabField(descriptionTextField);
descriptionTextField.setBacktabField(contactPhoneTextField);

}

public void setOwner(CustomInputPanel.CustomInputPanelOwner anOwner) {
    super.setOwner(anOwner);
    if (owner != null & !dataLoaded) {
        POST_URL_ADDRESS =
            (String)Interchange.panelPreferences.get(((CustomInputPanel.CustomInputPanelOwner)owner).getChannelName().toUpperCase()+".URL.POST");
        HELP_URL_ADDRESS =
            (String)Interchange.panelPreferences.get(((CustomInputPanel.CustomInputPanelOwner)owner).getChannelName().toUpperCase()+".URL.HELP");
        DATA_URL_ADDRESS =
            (String)Interchange.panelPreferences.get(((CustomInputPanel.CustomInputPanelOwner)owner).getChannelName().toUpperCase()+".URL.DATA");
        // DATA_URL_ADDRESS = "file:///C:/TEMP/cqpanel.xml";
        configurePopups();
    }
}

protected Vector configDisplayKeys(XMLPreferenceReader xml, String elementName, Popup popup) {
    TXElement popupArray[] = xml.getElementsNamed(elementName);
    if (popupArray == null)
        return null;
    popup.removeAllItems();
    //add "Pick one" on the first item to do sanity check
    popup.addItem(PICK_ONE, null);
    Vector returnVector = new Vector();
    returnVector.addElement("0");
    for (int i=0; i< popupArray.length; i++) {
        String display = popupArray[i].getElementNamed("Display").getText();
        String key = popupArray[i].getElementNamed("Key").getText();
        popup.addItem(display, null);
        returnVector.addElement(key);
    }
    return returnVector;
}

protected void configurePopups() {
    try {
        String response = handleHTTPPost(null, null, DATA_URL_ADDRESS, null);
        int start = response.indexOf("<?xml");
        int end = response.indexOf("</xml>");
        response = response.substring(start, end);
        System.out.println(response);
        StringReader reader = new StringReader(response);
        XMLPreferenceReader xml = new XMLPreferenceReader("xxx", (Reader)reader);
        categoryKeys = configDisplayKeys(xml, "CustomQCategory", categoryPopup);
        centerKeys = configDisplayKeys(xml, "CustomQtsc", centerPopup);
        typeKeys = configDisplayKeys(xml, "CustomQType", typePopup);
        scopeKeys = configDisplayKeys(xml, "CustomQScope", scopePopup);
        dataLoaded = true;
        xml.destroy();
        reader.close();
    } catch (Exception e) {
        Debug.report(this.getClass(), Debug.INFO_DEBUG, e);
        dataLoaded = false;
    }
}

```

```

CustomQInputPanel.java
// Post error message to its own
// channel window only (Bug0414 - Jiefei)
if (owner != null) {
    ((CustomInputPanel.CustomInputPanelOwner)owner).addMessage("Error loading Data: try
to switch back to this channel later");
}
}

public void resetPanelSize(int width, int height) {
    // make some calculations
    // int tfw = ( (width - 4*MARGIN) / 3 );
    int halfwidth = (width - 5*MARGIN)/ 2;
    int thirdwidth = (width - 8*MARGIN)/3;
    int yPos = MARGIN + typeLabel.height() + SMALL_MARGIN;
    int TEXT_HEIGHT = typeLabel.height() + 3;
    // layout the stuff in the container
    head1Label.moveTo(MARGIN, MARGIN);
    head1Label.setSize(width, TEXT_HEIGHT);
    head2Label.moveTo(MARGIN, yPos);
    head2Label.setSize(width, TEXT_HEIGHT);
    yPos += 2*(MARGIN + TEXT_HEIGHT);
    contactPhoneLabel.moveTo(MARGIN, yPos);
    contactPhoneLabel.setSize(contactPhoneLabel.minSize().width,
contactPhoneLabel.minSize().height);
    int xPos = MARGIN + halfwidth + 3*MARGIN;
    centerLabel.moveTo(xPos, yPos);
    centerLabel.setSize(centerLabel.minSize().width, centerLabel.minSize().height);
    yPos += SMALL_MARGIN + centerLabel.height();
    contactPhoneTextField.moveTo(MARGIN, yPos);
    contactPhoneTextField.setSize(halfwidth, TEXT_HEIGHT);
    centerPopup.moveTo(xPos, yPos);
    centerPopup.setSize(halfwidth, TEXT_HEIGHT);

    yPos += MARGIN + centerPopup.height();
    typeLabel.moveTo(MARGIN, yPos);
    xPos = MARGIN + thirdwidth + 3*MARGIN;
    typeLabel.setSize(typeLabel.minSize().width, typeLabel.minSize().height);
    categoryLabel.moveTo(xPos, yPos);
    categoryLabel.setSize(categoryLabel.minSize().width, categoryLabel.minSize().height);
    xPos += thirdwidth + 3*MARGIN;
    scopeLabel.moveTo(xPos, yPos);
    scopeLabel.setSize(scopeLabel.minSize().width, scopeLabel.minSize().height);

    yPos += SMALL_MARGIN + typeLabel.height();
    typePopup.moveTo(MARGIN, yPos);
    typePopup.setSize(thirdwidth, TEXT_HEIGHT);
    xPos = MARGIN + thirdwidth + 3*MARGIN;
    categoryPopup.moveTo(xPos, yPos);
    categoryPopup.setSize(thirdwidth, TEXT_HEIGHT);
    xPos += thirdwidth + 3*MARGIN;
    scopePopup.moveTo(xPos, yPos);
    scopePopup.setSize(thirdwidth, TEXT_HEIGHT);

    // yPos += MARGIN + centerPopup.height();
    // scopeLabel.moveTo(xPos, yPos);
    // scopeLabel.setSize(scopeLabel.minSize().width, scopeLabel.minSize().height);
    // yPos += SMALL_MARGIN + typeLabel.height();
    // scopePopup.moveTo(xPos, yPos);
    // scopePopup.setSize(halfwidth, TEXT_HEIGHT);
    yPos += MARGIN + centerPopup.height();
    descriptionLabel.moveTo(MARGIN, yPos);
    descriptionLabel.setSize(descriptionLabel.minSize().width,
descriptionLabel.minSize().height);
    yPos += descriptionLabel.height() + SMALL_MARGIN;
    descriptionTextField.moveTo(MARGIN, yPos);
    descriptionTextField.setSize(width-(2*MARGIN), TEXT_HEIGHT);
    yPos += descriptionTextField.height() + MARGIN;
    xPos = (width - 75*3 - 2*MARGIN) / 2;
}

```



```

                                CustomQInputPanel.java
submitButton.moveTo( xPos ,yPos);
submitButton.setSize(75, TEXT_HEIGHT + 5);
xPos += 75 + MARGIN;
clearButton.moveTo(xPos, yPos);
clearButton.setSize(75, TEXT_HEIGHT + 5);
xPos += 75 + MARGIN;
helpButton.moveTo(xPos ,yPos);
helpButton.setSize(75, TEXT_HEIGHT + 5);
}

protected void handlePanelClear() {
    // reset the colors
    contactPhoneLabel.setColor(Color.black);
    contactPhoneTextField.setStringValue("");
    centerLabel.setColor(Color.black);
    centerPopup.selectItemAt(0);
    typeLabel.setColor(Color.black);
    //sourceLabel.setColor(Color.black);
    descriptionLabel.setColor(Color.black);
    //priorityLabel.setColor(Color.black);
    scopeLabel.setColor(Color.black);
    categoryLabel.setColor(Color.black);
    // clear the text fields
    descriptionTextField.setStringValue("");
    typePopup.selectItemAt(0);
    //sourcePopup.selectItemAt(0);
    // priorityPopup.selectItemAt(0);
    scopePopup.selectItemAt(0);
    categoryPopup.selectItemAt(0);
    contactPhoneTextField.setFocusedView();
}

protected void handlePanelSubmit() {
    // reset the colors
    contactPhoneLabel.setColor(Color.black);
    centerLabel.setColor(Color.black);
    typeLabel.setColor(Color.black);
    //sourceLabel.setColor(Color.black);
    descriptionLabel.setColor(Color.black);
    // priorityLabel.setColor(Color.black);
    scopeLabel.setColor(Color.black);
    categoryLabel.setColor(Color.black);
    // get the inputs
    String contactPhone = contactPhoneTextField.stringValue();
    int centerSelected = centerPopup.selectedIndex();
    int typeSelected = typePopup.selectedIndex();
    int categorySelected = categoryPopup.selectedIndex();
    int scopeSelected = scopePopup.selectedIndex();
    String description = descriptionTextField.stringValue();

    // String source = sourcePopup.selectedItem().title();

    // String priority = priorityPopup.selectedItem().title();
    // String scope = scopePopup.selectedItem().title();
    // String category = categoryPopup.selectedItem().title();
    // check to see if everything is ok
    if (contactPhone.equals("")) {
        contactPhoneLabel.setColor(Color.red);
        contactPhoneTextField.setFocusedView();
    }
    else if (centerSelected == 0) {
        centerLabel.setColor(Color.red);
    }
    else if (typeSelected == 0) {
        typeLabel.setColor(Color.red);
    }
    else if (categorySelected == 0) {
        categoryLabel.setColor(Color.red);
    }
}

```

```

CustomQInputPanel.java
else if (scopeSelected == 0) {
    scopeLabel.setColor(Color.red);
}
else if (description.equals("")) {
    descriptionLabel.setColor(Color.red);
    descriptionTextField.setFocusedview();
}
else {
    // construct the strings
    String postData = "";
    // String message = "";
    // get some info
    InetAddress localhost = null;
    try { localhost = InetAddress.getLocalHost(); } catch (Exception e) {
        e.printStackTrace();
    }
    String machineName = localhost.getHostName();
    String machineIP = localhost.getHostAddress();
    // set up the message information
    preferences.InterchangeSettings is =
Interchange.applicationObject.getInterchangeSettings();
    Date now = new Date();

    postData = "Workstation="+URLLEncoder.encode(machineName)
        + "&username=" + URLLEncoder.encode(System.getProperty("user.name"))
        + "&ipnumber=" + URLLEncoder.encode(machineIP)
        + "&phone="+URLLEncoder.encode(contactPhone)
        + "&tsc=" + centerKeys.elementAt(centerSelected)
        + "&tz=" + URLLEncoder.encode(TimeZone.getDefault().getID())
        + "&Type=" + typeKeys.elementAt(typeSelected)
        + "&Category=" + categoryKeys.elementAt(categorySelected)
        + "&Scope=" + scopeKeys.elementAt(scopeSelected)
        + "&clientRequestTime=" + URLLEncoder.encode(now.toLocaleString())
        + "&clientRequestTimeGMT=" + URLLEncoder.encode(now.toGMTString())
        + "&chatUsername=" + URLLEncoder.encode(is.getRealname())
        + "&ProblemText=" + URLLEncoder.encode(description);

    /*
    message = "Net Customer Information " + '\n'
    + "\t\t Real Name: " + is.getRealname() + '\n'
    + "\t\t Email Address: " + is.getEmail() + '\n'
    + "\t\t NT Login: " + System.getProperty("user.name") + '\n'
    + "\t\t Machine Name: " + machineName + '\n'
    + "\t\t Machine IP: " + machineIP + '\n'
    + "\t\t OS Name: " + System.getProperty("os.name") + '\n'
    + "\t\t OS Architecture: " + System.getProperty("os.arch") + '\n'
    + "\t\t OS Version: " + System.getProperty("os.version") + '\n'
    + "\t\t New Ticket Information " + '\n'
    + "\t\t Contact Phone Number: " + contactPhone + '\n'
    + "\t\t Technical Support Center: " + centerDisplayItems[centerSelected] + '\n'
    + "\t\t Type: " + typeDisplayItems[typeSelected] + '\n'
    + "\t\t Category: " + categoryDisplayItems[categorySelected] + '\n'
    + "\t\t Scope: " + scopeDisplayItems[scopeSelected] + '\n'
    + "\t\t Description: " + description;

    if (owner != null) {
        ((CustomInputPanel.CustomInputPanelOwner)owner).messageEntered(message);
    }
    */

    // some debug info
    if (DEBUG) {
        System.out.println("CustomQ Server" + "\n " + POST_URL_ADDRESS);
        System.out.println("Message going to CustomQ" + "\n " + postData);
    }

    // perform the HTTP Post and retrieve the results
    String response = handleHTTPPost(null, null, POST_URL_ADDRESS, postData);
    if (owner != null) {
        StringTokenizer responseLines = new StringTokenizer(response, "\n");

```

```

CustomQInputPanel.java
while (responseLines.hasMoreTokens()) {
((CustomInputPanel.CustomInputPanelOwner)owner).messageEntered(responseLines.nextToken());
}
    if (DEBUG) {
        system.out.println("Response from CustomQ: " + response);
    }

    handlePanelClear();
    return;
}
Toolkit.getDefaultToolkit().beep();
}

protected void handlePanelHelp() {
    try {
        webUtilities.showDocument(HELP_URL_ADDRESS);
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}

public void setupFocus() {

    public void textEditingDidEnd(TextField tf, int endCondition, boolean
contentsChanged) {
        if (tf == descriptionTextField) {
            if (endCondition == TextFieldOwner.RETURN_KEY)
                submitButton.click();
        }
    }

    public void textEditingDidBegin(TextField tf) { }

    public boolean textEditingWillEnd(TextField tf, int endCondition, boolean
contentsChanged) { return true; }

    public void textWasModified(TextField tf) { }

    public void performCommand(String stCommand, Object object) {
        if (SUBMIT.equals(stCommand))
            handlePanelSubmit();
        else if (CLEAR.equals(stCommand))
            handlePanelClear();
        else if (HELP.equals(stCommand))
            handlePanelHelp();
    }
}

```

```
package ui.views.channel.custom;
```

```
import netscape.application.*;
import java.awt.Toolkit;
import java.net.*;
import java.io.*;

import core.Interchange;

import COM.swissbank.widgets.ifc.core.*;
import COM.swissbank.widgets.ifc.text.*;
import COM.swissbank.widgets.ifc.toolbar.*;
import COM.swissbank.widgets.ifc.buttons.*;
import COM.swissbank.widgets.ifc.splitter.*;
import COM.swissbank.util.TargetFacility;
```

```
public class EqInputPanel extends HTTPInputPanel implements Target,
                                                                    TextFieldOwner
```

```
{
    private static final String POST_LOGIN = "interchange1";
    private static final String POST_PASSWORD = "prodigy";
    public static String POST_URL_ADDRESS;

    protected final static String SUBMIT = "Submit";
    protected final static String CLEAR = "Clear";
    protected final static String AUTHOR = "Author";
    protected final static String SOURCE = "Source";
    protected final static String SYMBOL = "Symbol";
    protected final static String SECTOR = "Sector";
    protected final static String COMMENT = "Comment";
    protected final static String PICK_ONE = "Pick One";

    protected Label                                authorLabel;
    protected TextField                            authorTextField;
    protected Label                                sourceLabel;
    protected Popup                                sourcePopup;
    protected Label                                symbolLabel;
    protected TextField                            symbolTextField;
    protected Label                                sectorLabel;
    protected Popup                                sectorPopup;
    protected Label                                commentLabel;
    protected TextField                            commentTextField;
    protected Button                               clearButton;
    protected Button                               submitButton;

    public EqInputPanel(){
        super();
        PREFERRED_HEIGHT = 120;
    }

    public void setOwner(CustomInputPanel.CustomInputPanelOwner anOwner) {
        super.setOwner(anOwner);
        if (owner != null) {
            POST_URL_ADDRESS =
                (String)Interchange.panelPreferences.get(((CustomInputPanel.CustomInputPanelOwner)owner).getChannelName().toUpperCase()+".URL.POST");
        }
    }

    protected void createwidgets() {
        super.createwidgets();
        // create the author stuff
        authorLabel = new Label();
        authorLabel.setTitle(AUTHOR);
        authorLabel.setJustification(Graphics.RIGHT_JUSTIFIED);
        panelContainer.addSubview(authorLabel);
        authorTextField = new TextField();
        authorTextField.setOwner(this);
        authorTextField.setStringValue(System.getProperty("user.name"));
    }
}
```

```

authorTextField.setEditable(false);
authorTextField.setBackgroundColor(Color.lightGray);
panelContainer.addSubview(authorTextField);
// create the source stuff
sourceLabel = new Label();
sourceLabel.setTitle(SOURCE);
sourceLabel.setJustification(Graphics.RIGHT_JUSTIFIED);
panelContainer.addSubview(sourceLabel);
sourcePopup = new Popup();
sourcePopup.addItem(PICK_ONE, null);
sourcePopup.addItem("ALX", null);
sourcePopup.addItem("BS", null);
sourcePopup.addItem("C", null);
sourcePopup.addItem("DMG", null);
sourcePopup.addItem("DRW", null);
sourcePopup.addItem("DLJ", null);
sourcePopup.addItem("FB", null);
sourcePopup.addItem("FUR", null);
sourcePopup.addItem("GS", null);
sourcePopup.addItem("HQ", null);
sourcePopup.addItem("JPM", null);
sourcePopup.addItem("LEH", null);
sourcePopup.addItem("MER", null);
sourcePopup.addItem("MON", null);
sourcePopup.addItem("MS", null);
sourcePopup.addItem("OP", null);
sourcePopup.addItem("PW", null);
sourcePopup.addItem("PRU", null);
sourcePopup.addItem("ROB", null);
sourcePopup.addItem("SBC", null);
sourcePopup.addItem("SCB", null);
sourcePopup.addItem("SBH", null);
sourcePopup.addItem("SDV", null);
sourcePopup.addItem("WDR", null);
sourcePopup.addItem("WER", null);
sourcePopup.addItem("WB", null);
sourcePopup.addItem("JS", null);
sourcePopup.addItem("FT", null);
sourcePopup.addItem("NYT", null);
sourcePopup.addItem("DJ", null);
sourcePopup.addItem("WSJ", null);
sourcePopup.addItem("IBD", null);
sourcePopup.addItem("BAR", null);
sourcePopup.addItem("RT", null);
panelContainer.addSubview(sourcePopup);
// create the symbol stuff
symbolLabel = new Label();
symbolLabel.setTitle(SYMBOL);
symbolLabel.setJustification(Graphics.RIGHT_JUSTIFIED);
panelContainer.addSubview(symbolLabel);
symbolTextField = new TextField();
symbolTextField.setOwner(this);
panelContainer.addSubview(symbolTextField);
// create the risk class stuff
sectorLabel = new Label();
sectorLabel.setTitle(SECTOR);
sectorLabel.setJustification(Graphics.RIGHT_JUSTIFIED);
panelContainer.addSubview(sectorLabel);
sectorPopup = new Popup();
sectorPopup.addItem(PICK_ONE, null);
sectorPopup.addItem("CONSTRUCTION", null);
sectorPopup.addItem("CHEMICALS", null);
sectorPopup.addItem("CONSUMER", null);
sectorPopup.addItem("DEFENSE", null);
sectorPopup.addItem("DRUGS", null);
sectorPopup.addItem("DURABLES", null);
sectorPopup.addItem("ENERGY", null);
sectorPopup.addItem("EMTERTNMNT", null);
sectorPopup.addItem("ENVIRON_SRV", null);
sectorPopup.addItem("FINANCIAL", null);

```

```

EqInputPanel.java
sectorPopup.addItem("HEALTHCARE", null);
sectorPopup.addItem("HI_TECH", null);
sectorPopup.addItem("KITCHENSINK", null);
sectorPopup.addItem("MEDIA", null);
sectorPopup.addItem("METALS", null);
sectorPopup.addItem("REITS", null);
sectorPopup.addItem("RETAIL", null);
sectorPopup.addItem("TPHONES", null);
sectorPopup.addItem("TRANSPORTS", null);
sectorPopup.addItem("TREASURY", null);
sectorPopup.addItem("UNKNOWN", null);
sectorPopup.addItem("UTILITIES", null);
sectorPopup.addItem("WOODS", null);
panelContainer.addSubview(sectorPopup);
// create the comment stuff
commentLabel = new JLabel();
commentLabel.setTitle(COMMENT);
commentLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(commentLabel);
commentTextField = new TextField();
commentTextField.setOwner(this);
panelContainer.addSubview(commentTextField);
// create the clear button
clearButton = new Button();
clearButton.setTitle(CLEAR);
clearButton.setCommand(CLEAR);
clearButton.setTarget(this);
panelContainer.addSubview(clearButton);
// create the submit button
submitButton = new Button();
submitButton.setTitle(SUBMIT);
submitButton.setCommand(SUBMIT);
submitButton.setTarget(this);
panelContainer.addSubview(submitButton);
// setup the forward tabbing
symbolTextField.setTabField(commentTextField);
commentTextField.setTabField(symbolTextField);
// setup the back tabbing
symbolTextField.setBacktabField(commentTextField);
commentTextField.setBacktabField(symbolTextField);
}

public void resetPanelSize(int width, int height) {
// size all the labels to min size
authorLabel.setSize(authorLabel.minSize().width, authorLabel.minSize().height);
sourceLabel.setSize(sourceLabel.minSize().width, sourceLabel.minSize().height);
symbolLabel.setSize(symbolLabel.minSize().width, symbolLabel.minSize().height);
sectorLabel.setSize(sectorLabel.minSize().width, sectorLabel.minSize().height);
commentLabel.setSize(commentLabel.minSize().width, commentLabel.minSize().height);
// set them to same widths based on position
if (authorLabel.width() > symbolLabel.width()) {
    symbolLabel.setSize(authorLabel.minSize().width, symbolLabel.minSize().height);
}
else {
    authorLabel.setSize(symbolLabel.minSize().width, authorLabel.minSize().height);
}

if (sourceLabel.width() > sectorLabel.width()) {
    sectorLabel.setSize(sourceLabel.minSize().width, sectorLabel.minSize().height);
}
else {
    sourceLabel.setSize(sectorLabel.minSize().width, sourceLabel.minSize().height);
}
// make some calculations
int tfw = ((width - authorLabel.width() - 5*MARGIN) / 2);
int yPos = MARGIN + authorLabel.height();
int TEXT_HEIGHT = authorLabel.height() + 3;

// layout the stuff in the container
authorLabel.moveTo(MARGIN, MARGIN);

```

```

EqInputPanel.java
int xPos = 2*MARGIN + authorLabel.width();
authorTextField.moveTo(xPos, MARGIN);
authorTextField.setSize(tfw, TEXT_HEIGHT);
xPos += tfw + MARGIN;

sourceLabel.moveTo(xPos, MARGIN);
xPos += sourceLabel.width() + MARGIN;
sourcePopup.moveTo(xPos, MARGIN);
sourcePopup.setSize(tfw, TEXT_HEIGHT);

yPos += MARGIN;
symbolLabel.moveTo(MARGIN, yPos);
xPos = 2*MARGIN + symbolLabel.width();
symbolTextField.moveTo(xPos, yPos);
symbolTextField.setSize(tfw, TEXT_HEIGHT);
xPos += tfw + MARGIN;

sectorLabel.moveTo(xPos, yPos);
xPos += sectorLabel.width() + MARGIN;
sectorPopup.moveTo(xPos, yPos);
sectorPopup.setSize(tfw, TEXT_HEIGHT);

yPos += MARGIN + sectorPopup.height();

commentLabel.moveTo((int)((width / 2) - (.5 * commentLabel.width())) , yPos);
yPos += SMALL_MARGIN + commentLabel.height();
commentTextField.moveTo(MARGIN, yPos);
commentTextField.setSize(width - 2*MARGIN, height - yPos - (TEXT_HEIGHT + 5) - 2*MARGIN);

yPos += commentTextField.height() + MARGIN;
clearButton.moveTo( ((width / 2) - 75 - ((5*MARGIN)/2)) ,yPos);
clearButton.setSize(75, TEXT_HEIGHT + 5);
submitButton.moveTo( ((width / 2) + ((5*MARGIN)/2)) ,yPos);
submitButton.setSize(75, TEXT_HEIGHT + 5);
}

protected void handlePanelClear() {
    // reset the colors
    sourceLabel.setColor(Color.black);
    symbolLabel.setColor(Color.black);
    sectorLabel.setColor(Color.black);
    // clear the text fields
    sourcePopup.selectItemAt(0);
    symbolTextField.setStringValue("");
    sectorPopup.selectItemAt(0);
    commentTextField.setStringValue("");
    symbolLabel.setFocusedView();
}

protected void handlePanelSubmit() {
    // reset the colors
    sourceLabel.setColor(Color.black);
    symbolLabel.setColor(Color.black);
    sectorLabel.setColor(Color.black);
    // get the strings
    String author = authorTextField.stringValue();
    String source = sourcePopup.selectedItem().title();
    String symbol = symbolTextField.stringValue();
    String sector = sectorPopup.selectedItem().title();
    String comment = commentTextField.stringValue();
    // check to see if everything is ok
    if (source.equals(PICK_ONE)) {
        sourceLabel.setColor(Color.red);
    }
    else if (symbol.equals("")) {
        symbolLabel.setColor(Color.red);
        symbolTextField.setFocusedView();
    }
    else if (sector.equals(PICK_ONE)) {

```

```

EqInputPanel.java
    sectorLabel.setColor(Color.red);
}
else {
    // set up the post string
    String postData = "Author=" + URLEncoder.encode(author) +
        "&source=" + URLEncoder.encode(source) +
        "&symbols=" + URLEncoder.encode(symbol) +
        "&sector=" + URLEncoder.encode(sector);
    // if there is a comment send it
    if (comment != null)
        postData = postData + "&Comments=" + URLEncoder.encode(comment);
    // Perform the HTTP Post and retrieve the results
    String response = handleHTTPPost(POST_LOGIN, POST_PASSWORD,
POST_URL_ADDRESS, postData);
    if (DEBUG) {
        System.out.println("Response from Kaliedoscope: " +
response);
    }
    // output chat to channel
    if (owner != null) {
        ((CustomInputPanel.CustomInputPanelOwner)owner).messageEntered(symbol + " (" + source
+ ") " + comment);
    }
    symbolLabel.setFocusedView();
    handlePanelClear();
    return;
}
Toolkit.getDefaultToolkit().beep();
}

public void setupFocus() {

    public void textEditingDidEnd(TextField tf, int endCondition, boolean
contentsChanged) {
        if ((tf == authorTextField) || (tf == symbolTextField) || (tf == commentTextField)) {
            if (endCondition == TextFieldOwner.RETURN_KEY)
                submitButton.click();
        }
    }

    public void textEditingDidBegin(TextField tf) { }

    public boolean textEditingWillEnd(TextField tf, int endCondition, boolean
contentsChanged) { return true; }

    public void textWasModified(TextField tf) { }

    public void performCommand(String stCommand, Object object) {
        if (SUBMIT.equals(stCommand))
            handlePanelSubmit();
        else if (CLEAR.equals(stCommand))
            handlePanelClear();
    }
}

```



```

panelContainer.addSubview(commentScroll);

// create event date fields
eventDateLabel = new Label();
eventDateLabel.setTitle(EVENTDATELABEL);
eventDateLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(eventDateLabel);
// monthSepLabel = new Label();
// monthSepLabel.setTitle("/");
// monthSepLabel.setJustification(Graphics.CENTERED);
// panelContainer.addSubview(monthSepLabel);
// dateSepLabel = new Label();
// dateSepLabel.setTitle("/");
// dateSepLabel.setJustification(Graphics.CENTERED);
// panelContainer.addSubview(dateSepLabel);
// eventMonthTextField = new CutCopyPasteTextField();
// eventMonthTextField.setPopupMenuProvider(this);
// panelContainer.addSubview(eventMonthTextField);
// eventDateTextField = new CutCopyPasteTextField();
// eventDateTextField.setPopupMenuProvider(this);
// panelContainer.addSubview(eventDateTextField);
// eventYearTextField = new CutCopyPasteTextField();
// eventYearTextField.setPopupMenuProvider(this);
// panelContainer.addSubview(eventYearTextField);

eventDateTextField = new CutCopyPasteTextField();
eventDateTextField.setPopupMenuProvider(this);
panelContainer.addSubview(eventDateTextField);
// Date today = new Date();
//
eventDateTextField.setStringValue(today.getDate()+"/"+(today.getMonth()+1)+"/"+(today.getYear
()+1900));
eventDateTextField.setStringValue("mm/dd/yyyy");

// create the clear button
clearButton = new Button();
clearButton.setTitle(CLEAR);
clearButton.setCommand(CLEAR);
clearButton.setTarget(this);
panelContainer.addSubview(clearButton);
// create the submit button
submitButton = new Button();
submitButton.setTitle(SUBMIT);
submitButton.setCommand(SUBMIT);
submitButton.setTarget(this);
panelContainer.addSubview(submitButton);

// create url buttons
gotoLabel = new Label();
gotoLabel.setTitle(GOTO);
gotoLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(gotoLabel);
urlButtons = new Button[urlButtonLabels.length];
for (int i=0; i < urlButtonLabels.length; i++) {
    urlButtons[i] = new Button();
    urlButtons[i].setTitle(urlButtonLabels[i]);
    urlButtons[i].setCommand(urlButtonLabels[i]);
    urlButtons[i].setTarget(this);
    panelContainer.addSubview(urlButtons[i]);
}

// setup the forward tabbing
symbolTextField.setTabField(clientFindTextField);
clientFindTextField.setTabField(eventDateTextField);
eventDateTextField.setTabField(symbolTextField);
symbolTextField.setBacktabField(eventDateTextField);
clientFindTextField.setBacktabField(symbolTextField);
eventDateTextField.setBacktabField(clientFindTextField);

```

```

                                SellsideInputPanel.java
// priceTextField.setTabField(sizeTextField);
// sizeTextField.setTabField(commentTextField);
// commentTextField.setTabField(symbolTextField);
// setup the back tabbing
// symbolTextField.setBacktabField(commentTextField);
// priceTextField.setBacktabField(symbolTextField);
// sizeTextField.setBacktabField(priceTextField);
// commentTextField.setBacktabField(sizeTextField);
}

protected String SELLSIDE = "Sellside";
protected String KEYWORD = "KScopeKeyword";
protected String ITEM = "KScopeKeywordItem";
protected String NAME = "name";

protected void configurePopups() {
    try {
        // hit the server and get the page that has the popups
        String response = handleHTTPPost(POST_LOGIN, POST_PASSWORD, DATA_URL_ADDRESS, null);
        if (DEBUG) {
            System.out.println("Response from Kaliedoscope: " + response);
        }
        // massage the data, lotus puts crap in there
        int start = response.indexOf("<?xml");
        int end = response.indexOf("</xml>");
        response = response.substring(start, end);
        if (DEBUG) {
            System.out.println("Massaged data: " + response);
        }
        // create a stream from that String
        StringReader reader = new StringReader(response);
        // use the XML Preference reader to get out the data structure
        XMLPreferenceReader xml = new XMLPreferenceReader(SELLSIDE, (Reader)reader);
        // load the data
        TXElement popupArray[] = xml.getElementsNamed(KEYWORD);

        //clean up the list views/popup menus
        clientListView.removeAllItems();
        sectorPopup.removeAllItems();
        typeListView.removeAllItems();
        securityPopup.removeAllItems();
        priorityPopup.removeAllItems();
        copytoListView.removeAllItems();

        // parse the data and build the popups
        for (int i = 0; i < popupArray.length; i++) {
            // which popup is it
            String popup = ((TXElement)popupArray[i]).getAttribute(NAME);
            // get all the items and loop through them
            TXElement items[] = ((TXElement)popupArray[i]).searchDescendantsAll(ITEM);
            for (int j = 0; j < items.length; j++) {
                // get the text that should be on the popup
                String text = ((TXElement)items[j]).getText();
                if (CLIENT.equals(popup)) {
                    ListItem li = clientListView.addItem();
                    li.setTitle(text);
                    li.setFont(new Font(Font.defaultFont().name(), Font.PLAIN, 10));
                    System.out.println("added item:"+text+"\n");
                }
                else if (SECTOR.equals(popup)) {
                    ListItem li = sectorPopup.addItem(text, null);
                    li.setFont(new Font(Font.defaultFont().name(), Font.PLAIN, 10));
                }
                else if (ORDERTYPE.equals(popup)) {
                    ListItem li = typeListView.addItem();
                    li.setTitle(text);
                    li.setFont(new Font(Font.defaultFont().name(), Font.PLAIN, 10));
                }
                else if (SECURITY.equals(popup)) {
                    ListItem li = securityPopup.addItem(text, null);
                }
            }
        }
    }
}

```

```

        SellsideInputPanel.java
        li.setFont(new Font(Font.defaultFont().name(), Font.PLAIN, 10));
    }
    else if (PRIORITY.equals(popup)) {
        ListItem li = priorityPopup.addItem(text, null);
        li.setFont(new Font(Font.defaultFont().name(), Font.PLAIN, 10));
    }
    else if (COPYTO.equals(popup)) {
        ListItem li = copytoListView.addItem();
        li.setTitle(text);
        li.setFont(new Font(Font.defaultFont().name(), Font.PLAIN, 10));
    }
}
}
xml.destroy();
reader.close();
dataLoaded = true;
}
catch (Exception e) {
    Debug.report(this.getClass(), Debug.INFO_DEBUG, e);
    // post error message to its own
    // channel window only (Bug0414 - Jiefei)
    dataLoaded = false;
    if (owner != null) {
        ((CustomInputPanel.CustomInputPanelOwner)owner).addMessage("Error loading Data: try
to switch back to this channel later");
    }
}
}

public void resetPanelSize(int width, int height) {
    // make some calculations
    // int tfw = ( (width - 5*MARGIN) / 4 );

    int TEXT_HEIGHT = symbolLabel.height() + 3;
    int tfw = MARGIN;
    // int tfw = priceTextField.font().fontMetrics().stringwidth("#####") + MARGIN;
    // int rightTextFieldWidth = priceLabel.minSize().width + tfw + 2*MARGIN;
    int rightTextFieldWidth = 3*MARGIN;
    int symbolTotWidth = ((width - 2*MARGIN) - rightTextFieldWidth) * 6 / 10;
    int securityTotWidth = ((width - 2*MARGIN) - rightTextFieldWidth) * 4 / 10;

    // layout the stuff in the container

    // size for each field in the first two rows
    int f1w = symbolLabel.minSize().width;
    int f2w = symbolTotWidth - MARGIN - symbolLabel.minSize().width;
    int f3w = securityLabel.minSize().width;
    int f4w = securityTotWidth - securityLabel.minSize().width - 2*MARGIN;
    // int f5w = priceLabel.minSize().width;
    //first row
    symbolLabel.moveTo(MARGIN, MARGIN);
    symbolLabel.setSize(f1w, symbolLabel.minSize().height);
    int xPos = MARGIN + f1w + MARGIN;
    symbolTextField.moveTo(xPos, MARGIN);
    symbolTextField.setSize(f2w, TEXT_HEIGHT);
    xPos = MARGIN + symbolTotWidth + MARGIN;
    securityLabel.moveTo(xPos, MARGIN);
    securityLabel.setSize(f3w, securityLabel.minSize().height);
    xPos += f3w + MARGIN;
    securityPopup.moveTo(xPos, MARGIN);
    securityPopup.setSize(f4w, TEXT_HEIGHT);
    xPos += f4w + MARGIN;
    // priceLabel.moveTo(xPos, MARGIN);
    // priceLabel.setSize(f5w, priceLabel.minSize().height);
    // xPos += f5w + MARGIN;
    // priceTextField.moveTo(xPos, MARGIN);
    // priceTextField.setSize(tfw, TEXT_HEIGHT);
    // second row
    int yPos = MARGIN + TEXT_HEIGHT + MARGIN;

```

SellsideInputPanel.java

```

sectorLabel.moveTo(MARGIN, yPos);
sectorLabel.setSize(sectorLabel.minSize().width, sectorLabel.minSize().height);
xPos = MARGIN + f1w + MARGIN;
sectorPopup.moveTo(xPos, yPos);
sectorPopup.setSize(f2w, TEXT_HEIGHT);
xPos = MARGIN + symbolTotwidth + MARGIN;
priorityLabel.moveTo(xPos, yPos);
priorityLabel.setSize(priorityLabel.minSize().width, priorityLabel.minSize().height);
xPos += f3w + MARGIN;
priorityPopup.moveTo(xPos, yPos);
priorityPopup.setSize(f4w, TEXT_HEIGHT);
xPos += f4w + MARGIN;
// sizeLabel.moveTo(xPos, yPos);
// sizeLabel.setSize(tfw, TEXT_HEIGHT);
// xPos += f5w + MARGIN;
// sizeTextField.moveTo(xPos, yPos);
// sizeTextField.setSize(tfw, TEXT_HEIGHT);

// sizes for row three and four
int clientwidth = (width - 4 * MARGIN)/2;
int copytewidth = (width - 4 * MARGIN)/4;
int typewidth = (width - 4 * MARGIN)/4;

//third row
yPos += TEXT_HEIGHT + MARGIN;
xPos = MARGIN;
clientLabel.moveTo(xPos, yPos);
clientLabel.setSize(clientLabel.minSize().width, clientLabel.minSize().height);
xPos += MARGIN + clientLabel.minSize().width;
clientFindTextField.moveTo(xPos, yPos);
int cftw = clientwidth - 2*MARGIN - BUTTON_WIDTH - clientLabel.minSize().width;
clientFindTextField.setSize(cftw, TEXT_HEIGHT);
xPos += cftw + MARGIN;
clientFindButton.moveTo(xPos, yPos);
clientFindButton.setSize(BUTTON_WIDTH, TEXT_HEIGHT);
xPos = MARGIN + clientwidth + MARGIN;
copytoLabel.moveTo(xPos, yPos);
copytoLabel.setSize(copytoLabel.minSize().width, copytoLabel.minSize().height);
xPos += copytewidth + MARGIN;
typeLabel.moveTo(xPos, yPos);
typeLabel.setSize(typeLabel.minSize().width, typeLabel.minSize().height);

// fourth row
yPos += TEXT_HEIGHT + MARGIN;
xPos = MARGIN;
clientScroll.moveTo(xPos, yPos);
clientScroll.setSize(clientwidth, TEXT_HEIGHT*3);
clientListView.setSizeToMinSize();
clientListView.setSizeTo(clientScroll.scrollview().bounds().width,
clientListView.count()*clientListView.rowHeight());
xPos += clientwidth + MARGIN;
copytoScroll.moveTo(xPos, yPos);
copytoScroll.setSize(copytewidth, TEXT_HEIGHT*3);
copytoListView.setSizeToMinSize();
copytoListView.setSizeTo(copytoScroll.scrollview().bounds().width,
copytoListView.count()*copytoListView.rowHeight());
xPos += copytewidth + MARGIN;
typeScroll.moveTo(xPos, yPos);
typeScroll.setSize(typewidth, TEXT_HEIGHT*3);
typeListView.setSizeToMinSize();
typeListView.setSizeTo(typeScroll.scrollview().bounds().width,
typeListView.count()*copytoListView.rowHeight());

// fifth row
int commentLabelwidth = commentLabel.minSize().width;
yPos += 3*TEXT_HEIGHT + MARGIN;
xPos = MARGIN;
commentLabel.moveTo(xPos, yPos);
commentLabel.setSize(commentLabelwidth, commentLabel.minSize().height);
xPos += commentLabelwidth + MARGIN;

```

SellsideInputPanel.java

```

commentScroll.moveTo(xPos, yPos);
commentScroll.setSize(width-(3*MARGIN) - commentLabelwidth, TEXT_HEIGHT*3);
Rect contentBounds = commentScroll.scrollView().bounds();
commentTextField.setSizeToMinSize();
commentTextField.setSizeTo(contentBounds.width, contentBounds.height);
// commentTextField.moveTo(xPos, yPos);
// commentTextField.setSizeTo(width-(3*MARGIN) - commentLabelwidth, TEXT_HEIGHT*3);

// sixth row
int eventDateWidth = (width - 7*MARGIN)/2;
// int digSize = eventMonthTextField.font().fontMetrics().stringwidth("9");
yPos += 3*TEXT_HEIGHT + MARGIN;
xPos = MARGIN;
eventDateLabel.moveTo(xPos, yPos);
eventDateLabel.setSizeTo(eventDateLabel.minSize().width, eventDateLabel.minSize().height);
xPos += eventDateLabel.minSize().width + MARGIN;
// eventMonthTextField.moveTo(xPos, yPos);
// eventMonthTextField.setSizeTo(3*digSize, TEXT_HEIGHT);
// xPos += 3*digSize;
// monthSepLabel.moveTo(xPos, yPos);
// monthSepLabel.setSizeTo(monthSepLabel.minSize().width, monthSepLabel.minSize().height);
// xPos += monthSepLabel.minSize().width;
// eventDateTextField.moveTo(xPos, yPos);
// eventDateTextField.setSizeTo(3*digSize, TEXT_HEIGHT);
// xPos += 3*digSize;
// dateSepLabel.moveTo(xPos, yPos);
// dateSepLabel.setSizeTo(dateSepLabel.minSize().width, dateSepLabel.minSize().height);
// xPos += dateSepLabel.minSize().width;
// eventYearTextField.moveTo(xPos, yPos);
// eventYearTextField.setSizeTo(5*digSize, TEXT_HEIGHT);

eventDateTextField.moveTo(xPos, yPos);
eventDateTextField.setSizeTo(eventDateWidth - MARGIN - eventDateLabel.minSize().width,
TEXT_HEIGHT);
xPos = MARGIN + eventDateWidth + 5*MARGIN;

/*
xPos = width - MARGIN - BUTTON_WIDTH;
submitButton.moveTo(xPos, yPos);
submitButton.setSizeTo(BUTTON_WIDTH, TEXT_HEIGHT + 5);
xPos -= BUTTON_WIDTH + MARGIN;
clearButton.moveTo(xPos, yPos);
clearButton.setSizeTo(BUTTON_WIDTH, TEXT_HEIGHT + 5);
*/

clearButton.moveTo(xPos, yPos);
int buttonWidth = (eventDateWidth-3*MARGIN)/2;
clearButton.setSizeTo(buttonWidth, TEXT_HEIGHT + 5);
xPos += buttonWidth + 3*MARGIN;
submitButton.moveTo(xPos, yPos);
submitButton.setSizeTo(buttonWidth, TEXT_HEIGHT + 5);

// last row
yPos += TEXT_HEIGHT + 5 + MARGIN;
xPos = MARGIN;
gotoLabel.moveTo(xPos, yPos);
gotoLabel.setSizeTo(gotoLabel.minSize().width, gotoLabel.minSize().height);
buttonWidth = (width - gotoLabel.width() - 7 *MARGIN)/5;
xPos += gotoLabel.minSize().width + MARGIN;
for (int i=0; i < urlButtonLabels.length; i++) {
    urlButtons[i].moveTo(xPos, yPos);
    urlButtons[i].setSizeTo(buttonWidth, TEXT_HEIGHT+5);
    xPos += buttonWidth + MARGIN;
}
}
protected void handleSearchClient() {
    String searchString = clientFindTextField.stringValue().trim();
    // if (!searchString.trim().equals(""))
    // {

```

```

                                SellsideInputPanel.java
// need to escape searchString here
String response = handleHTTPPost(POST_LOGIN, POST_PASSWORD, findURL+"&search="+
URLCoder.encode(searchString) + "&user="+Interchange.applicationObject.getLogin(), null);
int start = response.indexOf("<?xml");
int end = response.indexOf("</xml>");
response = response.substring(start, end);
if (DEBUG) {
    System.out.println("Massaged data: " + response);
}
// create a stream from that String
StringReader reader = new StringReader(response);
// use the XML Preference reader to get out the data structure
XMLPreferenceReader xml = new XMLPreferenceReader(SELLSIDE, (Reader)reader);
// load the data
TXElement popupArray[] = xml.getElementsNamed(KEYWORD);
//clear up the client list
clientListView.removeAllItems();

for (int i = 0; i < popupArray.length; i++) {
    // which popup is it
    String popup = ((TXElement)popupArray[i]).getAttribute(NAME);
    if (CLIENT.equals(popup)) {
        // get all the items and loop through them
        TXElement items[] = ((TXElement)popupArray[i]).searchDescendantsAll(ITEM);
        for (int j = 0; j < items.length; j++) {
            String text = ((TXElement)items[j]).getText();
            ListItem li = clientListView.addItem();
            li.setTitle(text);
            li.setFont(new Font(Font.defaultFont().name(), Font.PLAIN, 10));
            System.out.println("added item:"+text+"\n");
        }
    }
}

clientListView.sizeToMinSize();
clientListView.sizeTo(clientScroll.scrollview().bounds().width,
clientListView.count()*clientListView.rowHeight());
clientScroll.setDirty(true);
// }
}

protected void handlePanelClear() {
    // reset the colors
    symbolLabel.setColor(Color.black);
    // sizeLabel.setColor(Color.black);
    // priceLabel.setColor(Color.black);
    typeLabel.setColor(Color.black);
    clientLabel.setColor(Color.black);
    sectorLabel.setColor(Color.black);
    priorityLabel.setColor(Color.black);
    // clear the text fields
    symbolTextField.setStringValue("");
    // sizeTextField.setStringValue("");
    // priceTextField.setStringValue("");
    // clientTextField.setStringValue("");
    // commentTextField.setStringValue("");
    typeListView.selectItemAt(0);
    clientListView.selectItemAt(0);
    sectorPopup.selectItemAt(0);
    priorityPopup.selectItemAt(0);
    securityPopup.selectItemAt(0);
    symbolTextField.setFocusedView();
    // Date today = new Date();
    //
    eventDateTextField.setStringValue(today.getDate()+"/"+(today.getMonth()+1)+"/"+(today.getYear()
    +1900));
    eventDateTextField.setStringValue("mm/dd/yyyy");
    commentTextField.setStringValue("");
    clientFindTextField.setStringValue("");
}

```

```

                                SellsideInputPanel.java
ListItem item = clientListView.itemAt(0);
clientListView.selectOnly(item);
clientListView.deselectItem(item);

item = copytoListView.itemAt(0);
copytoListView.selectOnly(item);
copytoListView.deselectItem(item);

item = typeListView.itemAt(0);
typeListView.selectOnly(item);
typeListView.deselectItem(item);

}

protected void handlePanelSubmit() {
    // reset the colors
    symbolLabel.setColor(Color.black);
    // sizeLabel.setColor(Color.black);
    // priceLabel.setColor(Color.black);
    typeLabel.setColor(Color.black);
    clientLabel.setColor(Color.black);
    sectorLabel.setColor(Color.black);
    priorityLabel.setColor(Color.black);
    securityLabel.setColor(Color.black);
    copytoLabel.setColor(Color.black);
    commentLabel.setColor(Color.black);
    eventDateLabel.setColor(Color.black);

    // get the strings
    String symbol = symbolTextField.stringValue();
    // String size = sizeTextField.stringValue();
    // String price = priceTextField.stringValue();
    Vector clientVector = clientListView.selectedItems();
    Vector typeVector = typeListView.selectedItems();
    Vector copytoVector = copytoListView.selectedItems();
    String sector = sectorPopup.selectedItem().title();
    String priority = priorityPopup.selectedItem().title();
    // String comment = commentTextField.stringValue();
    String comment = commentTextField.string();
    String security = securityPopup.selectedItem().title();
    String eventDate = eventDateTextField.stringValue();

    // construct the strings
    String postData = "";
    // String message = "";
    // set up the user information
    postData += "Author=" + Interchange.applicationObject.getLogin();
    // Set up the Symbol information
    if (!"".equals(symbol)) {
        // message += "<" + symbol + "> ";
        postData += "&Symbols=" + URLEncoder.encode(symbol);
    }

    // Set up the sector information
    if (!PICK_ONE.equals(sector)) {
        // message += sector + " ";
        postData += "&Sector=" + URLEncoder.encode(sector);
    }

    // Set up the security information
    if (!PICK_ONE.equals(security)) {
        // message += security + " ";
        postData += "&Security=" + URLEncoder.encode(security);
    }

    // Set up the proiority information
    if (!PICK_ONE.equals(priority)) {
        // message += priority + " ";
        postData += "&Priority=" + URLEncoder.encode(priority);
    }
}

```

```

    }

    // Set up the prices information
    // if (!"".equals(price)) {
    //     message += price + " ";
    //     postData += "&Price=" + URLEncoder.encode(price);
    // }

    // Set up the trade size information
    // if (!"".equals(size)) {
    //     message += size + " ";
    //     postData += "&Size=" + URLEncoder.encode(size);
    // }

    // Set up the client information
    if (clientVector.size() > 0) {
        postData += "&Client=";
        for (int vIndex = 0; vIndex < clientVector.size(); vIndex++) {
            String client = ((ListItem)clientVector.elementAt(vIndex)).title();
            // message += client + " ";
            postData += URLEncoder.encode(client);
            if (vIndex < clientVector.size()-1)
                postData += ";";
        }
    }

    // setup copy to information
    if (copytoVector.size() > 0) {
        postData += "&CopyTo=";
        for (int vIndex = 0; vIndex < copytoVector.size(); vIndex++) {
            String copyto = ((ListItem)copytoVector.elementAt(vIndex)).title();
            // message += copyto + " ";
            postData += URLEncoder.encode(copyto);
            if (vIndex < copytoVector.size()-1)
                postData += ";";
        }
    }

    // Set up the order type information
    if (typeVector.size() > 0) {
        postData += "&SubmissionType=";
        for (int vIndex = 0; vIndex < typeVector.size(); vIndex++) {
            String orderType = ((ListItem)typeVector.elementAt(vIndex)).title();
            // message += orderType + " ";
            postData += URLEncoder.encode(orderType);
            if (vIndex < typeVector.size()-1)
                postData += ";";
        }
    }

    // setup eventdate information
    if (!"".equals(eventDate)) {
        // message += eventDate;
        postData += "&EventDate=" + URLEncoder.encode(eventDate);
    }

    // Set up the comment information
    if (!"".equals(comment)) {
        // message += comment;
        postData += "&Comments=" + URLEncoder.encode(comment);
    }

    // set up the post string
    if (!"".equals(postData) && (postData.charAt(0) == '&'))
        postData = postData.substring(1);
    // some debug info
    if (DEBUG) {
        System.out.println("Kaleidoscope Server" + "\n " + POST_URL_ADDRESS);
        System.out.println("Message going to Kaleidoscope" + "\n " + postData);
        System.out.println("Message going to Chat" + "\n " + message);
    }
}

```



```

package ui.views.channel.custom;

import netscape.application.*;
import java.awt.Toolkit;
import java.net.*;
import java.io.*;
import java.util.*;

import core.Interchange;

import COM.swissbank.widgets.ifc.core.*;
import COM.swissbank.widgets.ifc.text.*;
import COM.swissbank.widgets.ifc.toolbar.*;
import COM.swissbank.widgets.ifc.buttons.*;
import COM.swissbank.widgets.ifc.splitter.*;
import COM.swissbank.util.TargetFacility;
import COM.swissbank.util.WebUtilities;

public class MDMInputPanel extends HTTPInputPanel implements Target, TextFieldOwner
{
    private String POST_URL_ADDRESS;
    private String VIEWSTATUS_URL_ADDRESS;

    protected final static String HEAD = "MDM Ticket Tool Entry";
    protected final static String SUBMIT = "Submit";
    protected final static String CLEAR = "Clear";
    protected final static String VIEWSTATUS = "View Status";
    protected final static String REALNAME = "Real Name:";
    protected final static String EMAIL = "Email:";
    protected final static String EMAILEG = "(e.g. Me.Name@wdr.com)";
    protected final static String PHONE = "Phone:";
    protected final static String PHONEEG = "(e.g. London x84444)";
    protected final static String TICKETTITLE = "Ticket Title:";
    protected final static String DETAIL = "Details of Problem:";
    protected final static String DETAIL1 = "(e.g. app name, hostname running on,";
    protected final static String DETAIL2 = "PMHOST using, and symbols being used)";

    protected Label headLabel;
    protected Label realNameLabel;
    protected CutCopyPasteTextField realNameTextField;
    protected Label emailLabel;
    protected Label emailEgLabel;
    protected CutCopyPasteTextField emailTextField;
    protected Label phoneLabel;
    protected Label phoneEgLabel;
    protected CutCopyPasteTextField phoneTextField;
    protected Label ticketTitleLabel;
    protected CutCopyPasteTextField ticketTitleTextField;
    protected Label detailLabel;
    protected Label detailEg1Label;
    protected Label detailEg2Label;
    protected CutCopyPasteTextField detailTextField;
    //protected CutCopyPasteTextView detailTextField;
    //protected ScrollGroup detailScroll;

    protected Button submitButton;
    protected Button clearButton;
    protected Button viewStatusButton;

    public MDMInputPanel(){
        super();
        PREFERRED_HEIGHT = 260;
    }

    protected void createwidgets() {
        super.createwidgets();
        //create headers

```

```

headLabel = new Label();
headLabel.setTitle(HEAD);
headLabel.setJustification(Graphics.CENTERED);
panelContainer.addSubview(headLabel);

//create contact stuff
realNameLabel = new Label();
realNameLabel.setTitle-REALNAME);
realNameLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(realNameLabel);
realNameTextField = new CutCopyPasteTextField();
realNameTextField.setOwner(this);
realNameTextField.setPopupMenuProvider(this);
panelContainer.addSubview(realNameTextField);

emailLabel = new Label();
emailLabel.setTitle(EMAIL);
emailLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(emailLabel);
emailEgLabel = new Label();
emailEgLabel.setTitle(EMAILLEG);
emailEgLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(emailEgLabel);
emailTextField = new CutCopyPasteTextField();
emailTextField.setOwner(this);
emailTextField.setPopupMenuProvider(this);
panelContainer.addSubview(emailTextField);

phoneLabel = new Label();
phoneLabel.setTitle(PHONE);
phoneLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(phoneLabel);
phoneEgLabel = new Label();
phoneEgLabel.setTitle(PHONELLEG);
phoneEgLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(phoneEgLabel);
phoneTextField = new CutCopyPasteTextField();
phoneTextField.setOwner(this);
phoneTextField.setPopupMenuProvider(this);
panelContainer.addSubview(phoneTextField);

// create ticket and detail
ticketTitleLabel = new Label();
ticketTitleLabel.setTitle(TICKETTITLE);
ticketTitleLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(ticketTitleLabel);
ticketTitleTextField = new CutCopyPasteTextField();
ticketTitleTextField.setOwner(this);
ticketTitleTextField.setPopupMenuProvider(this);
panelContainer.addSubview(ticketTitleTextField);

detailLabel = new Label();
detailLabel.setTitle(DETAIL);
detailLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(detailLabel);
detailEg1Label = new Label();
detailEg1Label.setTitle(DETAIL1);
detailEg1Label.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(detailEg1Label);
detailEg2Label = new Label();
detailEg2Label.setTitle(DETAIL2);
detailEg2Label.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(detailEg2Label);

//detailTextField = new CutCopyPasteTextView();
//detailTextField.setPopupMenuProvider(this);
//detailScroll = new ScrollGroup();
//detailScroll.setBuffered(true);
//detailScroll.setHorizScrollBarDisplay(ScrollGroup.NEVER_DISPLAY);
//detailScroll.setVertScrollBarDisplay(ScrollGroup.ALWAYS_DISPLAY);

```

```

MDMInputPanel.java
//detailScroll.setBackgroundColor(Color.white);
//detailScroll.setContentView(detailTextField);
//panelContainer.addSubview(detailScroll);

detailTextField = new CutCopyPasteTextField();
detailTextField.setOwner(this);
detailTextField.setPopupMenuProvider(this);
panelContainer.addSubview(detailTextField);

// create the submit button
submitButton = new Button();
submitButton.setTitle(SUBMIT);
submitButton.setCommand(SUBMIT);
submitButton.setTarget(this);
panelContainer.addSubview(submitButton);
// create the clear button
clearButton = new Button();
clearButton.setTitle(CLEAR);
clearButton.setCommand(CLEAR);
clearButton.setTarget(this);
panelContainer.addSubview(clearButton);
// create the view status button
viewStatusButton = new Button();
viewStatusButton.setTitle(VIEWSTATUS);
viewStatusButton.setCommand(VIEWSTATUS);
viewStatusButton.setTarget(this);
panelContainer.addSubview(viewStatusButton);

// setup the forward tabbing
realNameTextField.setTabField(emailTextField);
emailTextField.setTabField(phoneTextField);
phoneTextField.setTabField(ticketTitleTextField);
ticketTitleTextField.setTabField(detailTextField);
detailTextField.setTabField(realNameTextField);
// setup the back tabbing
realNameTextField.setBacktabField(detailTextField);
emailTextField.setBacktabField(realNameTextField);
phoneTextField.setBacktabField(emailTextField);
ticketTitleTextField.setBacktabField(phoneTextField);
detailTextField.setBacktabField(ticketTitleTextField);
}

public void setOwner(CustomInputPanel.CustomInputPanelOwner anOwner) {
    super.setOwner(anOwner);
    POST_URL_ADDRESS =
(String)Interchange.panelPreferences.get(((CustomInputPanel.CustomInputPanelOwner)owner).getChannelName().toUpperCase()+".URL.POST");
    VIEWSTATUS_URL_ADDRESS =
(String)Interchange.panelPreferences.get(((CustomInputPanel.CustomInputPanelOwner)owner).getChannelName().toUpperCase()+".URL.VIEWSTATUS");
}

protected void configurePopups() {

public void resetPanelSize(int width, int height) {
    Font boldFont = new Font(realNameLabel.font().name(), Font.BOLD,
realNameLabel.font().size());
    int fullwidth = width - 2 * MARGIN;
    int halfwidth = (width - 5*MARGIN)/ 2;
    int TEXT_HEIGHT = (height - 10) / 13 - SMALL_MARGIN;

    // layout the stuff in the container
    headLabel.moveTo(MARGIN, MARGIN);
    headLabel.setFont(boldFont);
    headLabel.setSizeTo(width, TEXT_HEIGHT);

    int yPos = MARGIN + TEXT_HEIGHT + SMALL_MARGIN;
    int xPos = MARGIN + halfwidth + 3*MARGIN;

```

MDMInputPanel.java

```

realNameLabel.moveTo(MARGIN, yPos);
realNameLabel.setFont(boldFont);
realNameLabel.setSize(halfwidth, TEXT_HEIGHT);

yPos += TEXT_HEIGHT + SMALL_MARGIN;
realNameTextField.moveTo(MARGIN, yPos);
realNameTextField.setSize(halfwidth, TEXT_HEIGHT);

yPos += TEXT_HEIGHT + SMALL_MARGIN;
emailLabel.moveTo(MARGIN, yPos);
emailLabel.setFont(boldFont);
emailLabel.setSize(halfwidth, TEXT_HEIGHT);
emailEgLabel.moveTo(MARGIN + emailLabel.minSize().width, yPos);
emailEgLabel.setSize(halfwidth - emailLabel.minSize().width, TEXT_HEIGHT);
phoneLabel.moveTo(xPos, yPos);
phoneLabel.setFont(boldFont);
phoneLabel.setSize(halfwidth, TEXT_HEIGHT);
phoneEgLabel.moveTo(xPos + phoneLabel.minSize().width, yPos);
phoneEgLabel.setSize(halfwidth - phoneLabel.minSize().width, TEXT_HEIGHT);

yPos += TEXT_HEIGHT + SMALL_MARGIN;
emailTextField.moveTo(MARGIN, yPos);
emailTextField.setSize(halfwidth, TEXT_HEIGHT);
phoneTextField.moveTo(xPos, yPos);
phoneTextField.setSize(halfwidth, TEXT_HEIGHT);

yPos += TEXT_HEIGHT + SMALL_MARGIN;
ticketTitleLabel.moveTo(MARGIN, yPos);
ticketTitleLabel.setFont(boldFont);
ticketTitleLabel.setSize(fullwidth, TEXT_HEIGHT);

yPos += TEXT_HEIGHT + SMALL_MARGIN;
ticketTitleTextField.moveTo(MARGIN, yPos);
ticketTitleTextField.setSize(fullwidth, TEXT_HEIGHT);

yPos += TEXT_HEIGHT + SMALL_MARGIN;
detailLabel.moveTo(MARGIN, yPos);
detailLabel.setFont(boldFont);
detailLabel.setSize(fullwidth, TEXT_HEIGHT);
detailEg1Label.moveTo(MARGIN + detailLabel.minSize().width, yPos);
detailEg1Label.setSize(fullwidth - detailLabel.minSize().width, TEXT_HEIGHT);

yPos += TEXT_HEIGHT + SMALL_MARGIN;
detailEg2Label.moveTo(MARGIN, yPos);
detailEg2Label.setSize(fullwidth, TEXT_HEIGHT);

yPos += TEXT_HEIGHT + SMALL_MARGIN;
//detailScroll.moveTo(MARGIN, yPos);
//detailScroll.setSize(fullwidth, 3 * TEXT_HEIGHT);
//Rect contentBounds = detailScroll.scrollView().bounds();
//detailTextField.setSizeToMinSize();
//detailTextField.setSizeTo(contentBounds.width, contentBounds.height);

detailTextField.moveTo(MARGIN, yPos);
detailTextField.setSize(fullwidth, TEXT_HEIGHT);

yPos += 3 * TEXT_HEIGHT + 2 * SMALL_MARGIN;
submitButton.moveTo(width/2 - 75 - 2 * SMALL_MARGIN, yPos);
submitButton.setFont(boldFont);
submitButton.setSize(75, TEXT_HEIGHT + 8);
clearButton.moveTo(width/2 + 2 * SMALL_MARGIN, yPos);
clearButton.setFont(boldFont);
clearButton.setSize(75, TEXT_HEIGHT + 8);
viewStatusButton.moveTo(width - 75 - 2 * SMALL_MARGIN, yPos);
viewStatusButton.setFont(boldFont);
viewStatusButton.setSize(75, TEXT_HEIGHT + 8);
}

```

```

protected void handlePanelClear() {
    // reset the colors
    realNameLabel.setColor(Color.black);
    emailLabel.setColor(Color.black);
    phoneLabel.setColor(Color.black);
    ticketTitleLabel.setColor(Color.black);
    detailLabel.setColor(Color.black);

    realNameTextField.setStringValue("");
    emailTextField.setStringValue("");
    phoneTextField.setStringValue("");
    ticketTitleTextField.setStringValue("");
    detailTextField.setStringValue("");
    //detailTextField.clear();

    realNameTextField.setFocusedView();
}

protected void handlePanelSubmit() {
    // reset the colors
    realNameLabel.setColor(Color.black);
    emailLabel.setColor(Color.black);
    phoneLabel.setColor(Color.black);
    ticketTitleLabel.setColor(Color.black);
    detailLabel.setColor(Color.black);

    String realName = realNameTextField.stringValue();
    String email = emailTextField.stringValue();
    String phone = phoneTextField.stringValue();
    String ticketTitle = ticketTitleTextField.stringValue();
    //String detail = detailTextField.string();
    String detail = detailTextField.stringValue();

    if (realName.equals("")) {
        realNameLabel.setColor(Color.red);
        realNameTextField.setFocusedView();
    }
    else if (email.equals("")) {
        emailLabel.setColor(Color.red);
        emailTextField.setFocusedView();
    }
    else if (phone.equals("")) {
        phoneLabel.setColor(Color.red);
        phoneTextField.setFocusedView();
    }
    else if (ticketTitle.equals("")) {
        ticketTitleLabel.setColor(Color.red);
        ticketTitleTextField.setFocusedview();
    }
    else if (detail.equals("")) {
        detailLabel.setColor(Color.red);
        detailTextField.setFocusedView();
    }
    else {
        // construct the strings
        String postData = "";
        postData = "?CT=" + URLEncoder.encode("#fsmmdm") + System.getProperty("user.name")
            + "&TL=" + URLEncoder.encode(ticketTitle)
            + "&CN=" + URLEncoder.encode(realName)
            + "&EM=" + URLEncoder.encode(email)
            + "&PN=" + URLEncoder.encode(phone)
            + "&PB=" + URLEncoder.encode(detail);

        // replace space with '+'
        StringBuffer postDataBuffer = new StringBuffer(postData);
        for (int i=0; i<postDataBuffer.length(); i++) {
            if (postDataBuffer.charAt(i) == ' ')
                postDataBuffer.setCharAt(i, '+');
        }
        // perform the HTTP Post and retrieve the results
    }
}

```

```

MDMInputPanel.java
String response = handleHTTPPost(null, null, POST_URL_ADDRESS,
postDataBuffer.toString());
if (owner != null) {
StringTokenizer responseLines = new StringTokenizer(response, "\n");
while (responseLines.hasMoreTokens()) {
((CustomInputPanel.CustomInputPanelOwner)owner).messageEntered(responseLines.nextToken());
}
}
if (DEBUG) {
System.out.println("Response from CustomQ: " + response);
}
handlePanelClear();
return;
}

}

protected void handlePanelViewStatus() {
try {
webUtilities.showDocument(VIEWSTATUS_URL_ADDRESS);
}
catch (Exception e) {
e.printStackTrace();
}
}

public void setupFocus() {

public void textEditingDidEnd(TextField tf, int endCondition, boolean
contentsChanged) {

public void textEditingDidBegin(TextField tf) { }

public boolean textEditingWillEnd(TextField tf, int endCondition, boolean
contentsChanged) { return true; }

public void textWasModified(TextField tf) { }

public void performCommand(String stCommand, Object object) {
if (SUBMIT.equals(stCommand))
handlePanelSubmit();
else if (CLEAR.equals(stCommand))
handlePanelClear();
else if (VIEWSTATUS.equals(stCommand))
handlePanelViewStatus();
}
}
}

```

```

package ui.views.channel.custom;

import netscape.application.*;
import java.awt.Toolkit;

import COM.swissbank.widgets.ifc.core.*;
import COM.swissbank.widgets.ifc.text.*;
import COM.swissbank.widgets.ifc.toolbar.*;
import COM.swissbank.widgets.ifc.buttons.*;
import COM.swissbank.widgets.ifc.splitter.*;
import COM.swissbank.util.TargetFacility;

public class OptstkInputPanel extends CustomInputPanel implements Target,
                                                                    TextFieldOwner
{
    protected final static String SUBMIT = "Submit";
    protected final static String CLEAR = "Clear";
    protected final static String SYMBOL = "Symbol";
    protected final static String PRICE = "Price";
    protected final static String QUANTITY = "Quantity";
    protected final static String COMMENT = "Comment";
    protected final static String RISKCLASS = "Risk Class";
    protected final static String BUY_SELL = "Buy/Sell";
    protected final static String BUY = "Buy";
    protected final static String SELL = "Sell";
    protected final static String POSITION = "Position";
    protected final static String LONG = "Long";
    protected final static String SHORT = "Short";
    protected final static String PICK_ONE = "Pick One";

    protected Label                                symbolLabel;
    protected TextField                            symbolTextField;
    protected Label                                priceLabel;
    protected TextField                            priceTextField;
    protected Label                                quantityLabel;
    protected TextField                            quantityTextField;
    protected Label                                riskClassLabel;
    protected Popup                               riskClassPopup;
    protected Label                                commentLabel;
    protected TextField                            commentTextField;
    protected Button                               clearButton;
    protected Button                               submitButton;
    protected Button                               buySellButton;
    protected Label                                buySellLabel;
    protected Button                               longShortButton;
    protected Label                                positionLabel;
    protected boolean                              longPosition = true;
    protected boolean                              buy = true;

    public OptstkInputPanel(){
        super();
        PREFERRED_HEIGHT = 118;
    }

    protected void createwidgets() {
        super.createwidgets();
        // create the symbol stuff
        symbolLabel = new Label();
        symbolLabel.setTitle(SYMBOL);
        symbolLabel.setJustification(Graphics.LEFT_JUSTIFIED);
        panelContainer.addSubview(symbolLabel);
        symbolTextField = new TextField();
        symbolTextField.setOwner(this);
        panelContainer.addSubview(symbolTextField);
        // create the pricing stuff
        priceLabel = new Label();
        priceLabel.setTitle(PRICE);
        priceLabel.setJustification(Graphics.LEFT_JUSTIFIED);
    }

```



```

OptstkInputPanel.java
panelContainer.addSubview(priceLabel);
priceTextField = new TextField();
priceTextField.setOwner(this);
panelContainer.addSubview(priceTextField);
// create the quantity stuff
quantityLabel = new Label();
quantityLabel.setTitle(QUANTITY);
quantityLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(quantityLabel);
quantityTextField = new TextField();
quantityTextField.setOwner(this);
panelContainer.addSubview(quantityTextField);
// create the comment stuff
commentLabel = new Label();
commentLabel.setTitle(COMMENT);
commentLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(commentLabel);
commentTextField = new TextField();
commentTextField.setOwner(this);
panelContainer.addSubview(commentTextField);
// create the risk class stuff
riskClassLabel = new Label();
riskClassLabel.setTitle(RISKCLASS);
riskClassLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(riskClassLabel);
riskClassPopup = new Popup();
riskClassPopup.addItem(PICK_ONE, null);
riskClassPopup.addItem("OPT", null);
riskClassPopup.addItem("PFA", null);
riskClassPopup.addItem("FR", null);
riskClassPopup.addItem("CAN", null);
riskClassPopup.addItem("SAG", null);
riskClassPopup.addItem("Y2K", null);
riskClassPopup.addItem("EVT", null);
riskClassPopup.addItem("CVT", null);
riskClassPopup.addItem("RA", null);
riskClassPopup.addItem("IDX", null);
riskClassPopup.addItem("CEF", null);
riskClassPopup.addItem("SEC", null);
riskClassPopup.addItem("RES", null);
panelContainer.addSubview(riskClassPopup);
// create the clear button
clearButton = new Button();
clearButton.setTitle(CLEAR);
clearButton.setCommand(CLEAR);
clearButton.setTarget(this);
panelContainer.addSubview(clearButton);
// create the submit button
submitButton = new Button();
submitButton.setTitle(SUBMIT);
submitButton.setCommand(SUBMIT);
submitButton.setTarget(this);
panelContainer.addSubview(submitButton);
//create position label
buySellLabel = new Label();
buySellLabel.setTitle(BUY_SELL);
buySellLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(buySellLabel);
// create the buy/sell button
buySellButton = new Button();
buySellButton.setTitle(BUY);
buySellButton.setCommand(BUY);
buySellButton.setTarget(this);
panelContainer.addSubview(buySellButton);
//create position label
positionLabel = new Label();
positionLabel.setTitle(POSITION);
positionLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(positionLabel);
// create the position button

```

```

                                OptstkInputPanel.java
longShortButton = new Button();
longShortButton.setTitle(LONG);
longShortButton.setCommand(LONG);
longShortButton.setTarget(this);
panelContainer.addSubview(longShortButton);

// setup the forward tabbing
symbolTextField.setTabField(priceTextField);
priceTextField.setTabField(quantityTextField);
quantityTextField.setTabField(commentTextField);
commentTextField.setTabField(symbolTextField);
// setup the back tabbing
symbolTextField.setBacktabField(commentTextField);
priceTextField.setBacktabField(symbolTextField);
quantityTextField.setBacktabField(priceTextField);
commentTextField.setBacktabField(quantityTextField);
}

public void resetPanelSize(int width, int height) {
    // make some calculations
    int tfw = ( (width - 5*MARGIN) / 4 );
    int yPos = MARGIN + symbolLabel.height() + SMALL_MARGIN;
    int TEXT_HEIGHT = symbolLabel.height() + 3;
    // layout the stuff in the container
    buySellLabel.moveTo(MARGIN,MARGIN);
    buySellLabel.setSize(buySellLabel.minSize().width, buySellLabel.minSize().height);
    buySellButton.moveTo(MARGIN, yPos);
    buySellButton.setSize(tfw, TEXT_HEIGHT);
    int xPos = tfw + 2*MARGIN;
    symbolLabel.moveTo(xPos, MARGIN);
    symbolLabel.setSize(symbolLabel.minSize().width, symbolLabel.minSize().height);
    symbolTextField.moveTo(xPos, yPos);
    symbolTextField.setSize(tfw, TEXT_HEIGHT);
    xPos += tfw + MARGIN;
    priceLabel.moveTo(xPos, MARGIN);
    priceLabel.setSize(priceLabel.minSize().width, priceLabel.minSize().height);
    priceTextField.moveTo(xPos, yPos);
    priceTextField.setSize(tfw, TEXT_HEIGHT);
    xPos += tfw + MARGIN;
    quantityLabel.moveTo(xPos, MARGIN);
    quantityLabel.setSize(quantityLabel.minSize().width, quantityLabel.minSize().height);
    quantityTextField.moveTo(xPos, yPos);
    quantityTextField.setSize(tfw, TEXT_HEIGHT);
    yPos += MARGIN + quantityTextField.height();
    xPos = MARGIN;
    tfw = ( (width - 4*MARGIN - buySellButton.width()) / 2 );
    positionLabel.moveTo(xPos, yPos);
    positionLabel.setSize(positionLabel.minSize().width, positionLabel.minSize().height);
    xPos += buySellButton.width()+MARGIN;
    riskClassLabel.moveTo(xPos, yPos);
    riskClassLabel.setSize(riskClassLabel.minSize().width, riskClassLabel.minSize().height);
    xPos += MARGIN + tfw;
    commentLabel.moveTo(xPos, yPos);
    commentLabel.setSize(commentLabel.minSize().width, commentLabel.minSize().height);
    xPos = MARGIN;
    yPos += symbolLabel.height() + SMALL_MARGIN;
    longShortButton.moveTo(xPos, yPos);
    longShortButton.setSize(buySellButton.width(),TEXT_HEIGHT);
    xPos += MARGIN + buySellButton.width();
    riskClassPopup.moveTo(xPos, yPos);
    riskClassPopup.setSize(tfw, TEXT_HEIGHT);
    xPos += MARGIN + tfw;
    commentTextField.moveTo(xPos, yPos);
    commentTextField.setSize(tfw, TEXT_HEIGHT);
    yPos += commentTextField.height() + MARGIN;
    clearButton.moveTo( ((width / 2) - 75 - ((5*MARGIN)/2)) ,yPos);
    clearButton.setSize(75, TEXT_HEIGHT + 5);
    submitButton.moveTo( ((width / 2) + ((5*MARGIN)/2)) ,yPos);
    submitButton.setSize(75, TEXT_HEIGHT + 5);
}

```

```

protected void handlePanelClear() {
    // reset the colors
    symbolLabel.setColor(Color.black);
    priceLabel.setColor(Color.black);
    quantityLabel.setColor(Color.black);
    riskClassLabel.setColor(Color.black);
    // clear the text fields
    symbolTextField.setStringValue("");
    priceTextField.setStringValue("");
    quantityTextField.setStringValue("");
    commentTextField.setStringValue("");
    riskClassPopup.selectItemAt(0);
    symbolTextField.setFocusedView();
}

protected void handlePanelSubmit() {
    // reset the colors
    symbolLabel.setColor(Color.black);
    priceLabel.setColor(Color.black);
    quantityLabel.setColor(Color.black);
    riskClassLabel.setColor(Color.black);
    // get the strings
    String symbol = symbolTextField.stringValue();
    String price = priceTextField.stringValue();
    String quantity = quantityTextField.stringValue();
    String riskClass = riskClassPopup.selectedItem().title();
    String comment = commentTextField.stringValue();
    // check to see if everything is ok

    if (symbol.equals("")) {
        symbolLabel.setColor(Color.red);
        symbolTextField.setFocusedView();
    }
    else if (price.equals("")) {
        priceLabel.setColor(Color.red);
        priceTextField.setFocusedView();
    }
    else if (quantity.equals("")) {
        quantityLabel.setColor(Color.red);
        quantityTextField.setFocusedView();
    }
    else if (riskClass.equals(PICK_ONE)) {
        riskClassLabel.setColor(Color.red);
    }
    else {
        if (((comment.equals("")) && (owner != null)) {
            ((CustomInputPanel.CustomInputPanelOwner)owner).messageEntered(getBuySellString() +
            quantity + " <" + symbol + "> at " + price + " for " + riskClass);
        }
        else {
            ((CustomInputPanel.CustomInputPanelOwner)owner).messageEntered(getBuySellString() + quantity
            + " <" + symbol + "> at " + price + " for " + riskClass + ", " + comment);
        }
        symbolTextField.setFocusedView();
        handlePanelClear();
        return;
    }
    Toolkit.getDefaultToolkit().beep();
}

public void setupFocus() {

    public void textEditingDidEnd(TextField tf, int endCondition, boolean
    contentsChanged) {
        if ((tf == symbolTextField) || (tf == priceTextField) || (tf == quantityTextField) ||
        (tf == commentTextField)) {

```

```

                                OptstkInputPanel.java
        if (endCondition == TextFieldOwner.RETURN_KEY)
            submitButton.click();
    }

    public void textEditingDidBegin(TextField tf) { }

    public boolean textEditingWillEnd(TextField tf, int endCondition, boolean
contentsChanged) { return true; }

    public void textWasModified(TextField tf) { }

    public void performCommand(String stCommand, Object object) {
        if (SUBMIT.equals(stCommand))
            handlePanelSubmit();
        else if (CLEAR.equals(stCommand))
            handlePanelClear();
        else if (BUY.equals(stCommand))
            handleToggleBuySellButton();
        else if (SELL.equals(stCommand))
            handleToggleBuySellButton();
        else if (LONG.equals(stCommand))
            handleToggleLongShortButton();
        else if (SHORT.equals(stCommand))
            handleToggleLongShortButton();
    }

    public void handleToggleLongShortButton() {
        longPosition = !longPosition;
        if(longPosition) {
            longShortButton.setTitle(LONG);
            longShortButton.setCommand(LONG);
        } else {
            longShortButton.setTitle(SHORT);
            longShortButton.setCommand(SHORT);
        }
    }

    public void handleToggleBuySellButton() {
        buy = !buy;
        if(buy) {
            buySellButton.setTitle(BUY);
            buySellButton.setCommand(BUY);
        } else {
            buySellButton.setTitle(SELL);
            buySellButton.setCommand(SELL);
        }
    }

    public String getBuySellString() {
        return (buy ? "+" : "-");
    }
}

```

```

package ui.views.channel;

import ui.controllers.*;
import ui.controllers.channel.*;
import netscape.application.*;
import netscape.util.Vector;

import COM.swissbank.widgets.ifc.core.*;
import COM.swissbank.widgets.ifc.text.*;
import COM.swissbank.widgets.ifc.toolbar.*;
import COM.swissbank.widgets.ifc.buttons.*;
import COM.swissbank.widgets.ifc.splitter.*;
import COM.swissbank.util.TargetFacility;
import COM.swissbank.util.webUtilities;

import core.*;
import preferences.*;

/**
 * @(#)PrivateChannelWindowView.java
 *
 * PrivateChannelWindowView is the main view that users will use when doing
 * person to person chatting.
 *
 * Copyright 1998 Warburg Dillon Read. All Rights Reserved.
 *
 * @author Don Bora, Sean Willson
 */
public class PrivateChannelWindowView extends ChannelWindowView {
    private static final String PRIVATE_CHANNEL_ICON = "private_small.gif";

    // overridden from superclass
    public void applyChannelSettings() {
        super.applyChannelSettings();
        //toolbarSet.setUserListEnabled(false);
        toolbarSet.setUserListShowing(false);
    }

    public Bitmap getChannelBitmap(){
        return webUtilities.findBitmap(PRIVATE_CHANNEL_ICON);
    }

    //public void setOwningUser(User _user){
    //    ((ChannelWindowViewMenu)
    toolbarSet.getMenu()).menuChannelProperties.setEnabled(false);
    //}

    public boolean getKickEnabled() {
        return false;
    }

    public boolean getMakeOperatorEnabled() {
        return false;
    }

    public boolean getChannelPropertiesEnabled() {
        return false;
    }

    //public void setShowJoinPart(boolean set) {}

```

PrivateChannelWindowView.java

```
public boolean getShowJoinPart() {
    return false;
}

public String getSelectedItem() {
    return (String)null;
}

    public boolean hasMemberList(){
        return false;
    }

/**
 *      Overridden from CustomView. Called from the constructor when the widgets
 *      need to be created. Note how here they are created and configured, but not
 *      positioned or sized.
 */
protected void createWidgets() {
    super.createWidgets();
    // removed by DSZ because it's stupid. probably an andy k. request.
}

//((ChannelWindowViewToolbar)this.toolbarSet.getToolbar()).removeUserListToggle();

// always force user list off
public boolean getShowUserListEnabled() {
    return false;
}

// added DSZ 4/13/99 - override from superclass to always force to false. you cannot have
// a member list on a private channel.
public void setShowUserList(boolean show) {
    this.getChannelSettings().setShowUserList(new Boolean(false));
    toolbarSet.setUserListShowing(false);
    resetSizes();
    setDirty(true);
    //if (show)
    // System.err.println("Cannot show user list in private chat window!");
}

public boolean getShowUserList() {
    return false;
}

// these aren't accessible from a Private channel
public void handleMakeOperator() {}
public void handleKick() {}
public void handleInvite(Channel channel) {}
}
```

```
package ui.views.channel.custom;
```

```
import netscape.application.*;
import com.ibm.xml.parser.*;
```

```
import java.awt.Toolkit;
import java.net.*;
import java.io.*;
import java.util.Date;
import netscape.util.*;
```

```
import core.Interchange;
import preferences.xml.*;
```

```
import COM.swissbank.widgets.ifc.core.*;
import COM.swissbank.widgets.ifc.text.*;
import COM.swissbank.widgets.ifc.toolbar.*;
import COM.swissbank.widgets.ifc.buttons.*;
import COM.swissbank.widgets.ifc.splitter.*;
import COM.swissbank.widgets.ifc.lists.*;
import COM.swissbank.util.TargetFacility;
import ui.views.channel.MessageEntryTextView;
import util.Debug;
import COM.swissbank.util.webUtilities;
```

```
public class SellsideInputPanel extends HTTPInputPanel implements Target /*,
                                                                    TextFieldOwner */
```

```
{
    public static final boolean DEBUG = true;

    private static final String POST_LOGIN = "interchange1";
    private static final String POST_PASSWORD = "prodigy";
    private String POST_URL_ADDRESS;
    private String DATA_URL_ADDRESS;
    private String findURL;
    private boolean dataLoaded = false;
    private String linkURLs[];

    protected final static int BUTTON_WIDTH = 50;
    protected final static String SUBMIT = "Submit";
    protected final static String CLEAR = "Clear";
    protected final static String SYMBOL = "Symbol";
    // protected final static String SIZE = "Size";
    // protected final static String PRICE = "Price";
    protected final static String ORDERTYPE = "SubmissionType";
    protected final static String ORDERTYPELABEL = "Submission Type";
    protected final static String CLIENT = "Client";
    protected final static String SECTOR = "Sector";
    protected final static String PRIORITY = "Priority";
    protected final static String SECURITY = "Security";
    protected final static String COPYTO = "CopyTo";
    protected final static String COPYTOLABEL = "Copy To";
    protected final static String COMMENT = "Comment";
    protected final static String PICK_ONE = "Pick One";
    protected final static String HIGH = "High";
    protected final static String HIGH_SYMBOL = "(H)";
    protected final static String EVENTDATE = "EventDate";
    protected final static String EVENTDATELABEL = "Event Date";
    protected final static String FIND = "Find";
    protected final static String GOTO = "Go to ...";
    protected final static String MORNINGREPORT = "Morning Report";
    protected final static String MYFAVORITES = "My Favorites";
    protected final static String SEARCH = "Search";
    protected final static String NEWCLIENT = "New Client";
    protected final static String FEEDBACK = "Feedback";
    protected final static String urlButtonLabels[] = {MORNINGREPORT, MYFAVORITES, SEARCH,
NEWCLIENT, FEEDBACK};
    protected final static String urlLinkKeys[] = { "MorningReport", "MyFavorites", "Search",
```

```

"NewClient", "Feedback"};

protected Label                                  symbolLabel;
protected CutCopyPasteTextField                 symbolTextField;
// protected Label                               sizeLabel;
// protected CutCopyPasteTextField               sizeTextField;
// protected Label                               priceLabel;
// protected CutCopyPasteTextField               priceTextField;
protected Label                               eventDateLabel;
// protected Label                               monthSepLabel;
// protected Label                               dateSepLabel;
// protected CutCopyPasteTextField eventMonthTextField;
// protected CutCopyPasteTextField eventDateTextField;
// protected CutCopyPasteTextField eventYearTextField;
protected CutCopyPasteTextField               eventDateTextField;

protected Label                                  clientLabel;
protected CutCopyPasteTextField                 clientFindTextField;
protected Button                               clientFindButton;
protected ScrollGroup                           clientScroll;
protected ExtendedListView                     clientListView;
protected Label                                  typeLabel;
protected ExtendedListView                     typeListView;
protected ScrollGroup                           typeScroll;
protected Label                                  copytoLabel;
protected ExtendedListView                     copytoListView;
protected ScrollGroup                           copytoScroll;

protected Label                                  sectorLabel;
protected Popup                               sectorPopup;
protected Label                                  priorityLabel;
protected Popup                               priorityPopup;
protected Popup                               securityPopup;
protected Label                                  securityLabel;

protected Label                                  commentLabel;
//protected CutCopyPasteTextField               commentTextField;
protected CutCopyPasteTextView                 commentTextFields;
protected ScrollGroup                           commentScroll;
protected Button                               clearButton;
protected Button                               submitButton;

protected Label                                  gotoLabel;
protected Button                               urlButtons[];

public SellsideInputPanel(){
    super();
    PREFERRED_HEIGHT = 250;
}

public void setOwner(CustomInputPanel.CustomInputPanelOwner anOwner) {
    super.setOwner(anOwner);
    if (owner != null && !dataLoaded) {
        linkURLs = new String[urlButtonLabels.length];
        for (int i=0; i< urlButtonLabels.length; i++) {
            linkURLs[i] =
(String)Interchange.panelPreferences.get(((CustomInputPanel.CustomInputPanelOwner)owner).getChannelName().toUpperCase()+".URL."+urlLinkKeys[i].toUpperCase());
        }

        findURL =
(String)Interchange.panelPreferences.get(((CustomInputPanel.CustomInputPanelOwner)owner).getChannelName().toUpperCase()+".URL.SEARCHCLIENT");
        POST_URL_ADDRESS =
(String)Interchange.panelPreferences.get(((CustomInputPanel.CustomInputPanelOwner)owner).getChannelName().toUpperCase()+".URL.POST");
        DATA_URL_ADDRESS =
(String)Interchange.panelPreferences.get(((CustomInputPanel.CustomInputPanelOwner)owner).getChannelName().toUpperCase()+".URL.DATA");
    }
}

```



```

                                SellsideInputPanel.java
channelName().toUpperCase()+".URL.DATA")+"&user="+Interchange.applicationObject.getLogin();
    dataLoaded = true;
    configurePopups();
}
}

protected void createwidgets() {
    super.createwidgets();
    // create the symbol stuff
    symbolLabel = new Label();
    symbolLabel.setTitle(SYMBOL);
    symbolLabel.setJustification(Graphics.LEFT_JUSTIFIED);
    panelContainer.addSubview(symbolLabel);
    symbolTextField = new CutCopyPasteTextField();
    // symbolTextField.setOwner(this);
    symbolTextField.setPopupMenuProvider(this);
    panelContainer.addSubview(symbolTextField);

    // create the size stuff
    // sizeLabel = new Label();
    // sizeLabel.setTitle(SIZE);
    // sizeLabel.setJustification(Graphics.LEFT_JUSTIFIED);
    // panelContainer.addSubview(sizeLabel);
    // sizeTextField = new CutCopyPasteTextField();
    // sizeTextField.setOwner(this);
    // sizeTextField.setPopupMenuProvider(this);
    // panelContainer.addSubview(sizeTextField);
    // create the pricing stuff
    // priceLabel = new Label();
    // priceLabel.setTitle(PRICE);
    // priceLabel.setJustification(Graphics.LEFT_JUSTIFIED);
    // panelContainer.addSubview(priceLabel);
    // priceTextField = new CutCopyPasteTextField();
    // priceTextField.setOwner(this);
    // priceTextField.setPopupMenuProvider(this);
    // panelContainer.addSubview(priceTextField);

    // create the type stuff
    typeLabel = new Label();
    typeLabel.setTitle(ORDERTYPELABEL);
    typeLabel.setJustification(Graphics.LEFT_JUSTIFIED);
    panelContainer.addSubview(typeLabel);
    typeListView = new ExtendedListView();
    typeListView.setAllowsEmptySelection(true);
    typeListView.setAllowsMultipleSelection(true);
    typeScroll = new ScrollGroup();
    typeScroll.setBuffered(true);
    typeScroll.setHorizScrollBarDisplay(ScrollGroup.NEVER_DISPLAY);
    typeScroll.setVertScrollBarDisplay(ScrollGroup.ALWAYS_DISPLAY);
    typeScroll.setContentView(typeListView);
    panelContainer.addSubview(typeScroll);

    // create the copyto stuff
    copytoLabel = new Label();
    copytoLabel.setTitle(COPYTOLABEL);
    copytoLabel.setJustification(Graphics.LEFT_JUSTIFIED);
    panelContainer.addSubview(copytoLabel);
    copytoListView = new ExtendedListView();
    copytoListView.setAllowsEmptySelection(true);
    copytoListView.setAllowsMultipleSelection(true);
    copytoScroll = new ScrollGroup();
    copytoScroll.setBuffered(true);
    copytoScroll.setHorizScrollBarDisplay(ScrollGroup.NEVER_DISPLAY);
    copytoScroll.setVertScrollBarDisplay(ScrollGroup.ALWAYS_DISPLAY);
    copytoScroll.setContentView(copytoListView);
    panelContainer.addSubview(copytoScroll);

    // create the client stuff
    clientLabel = new Label();
    clientLabel.setTitle(CLIENT);

```

```

                                SellsideInputPanel.java
clientLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(clientLabel);
clientFindTextField = new CutCopyPasteTextField();
clientFindTextField.setPopupMenuProvider(this);
panelContainer.addSubview(clientFindTextField);
clientFindButton = new Button();
clientFindButton.setTitle(FIND);
clientFindButton.setCommand(FIND);
clientFindButton.setTarget(this);
panelContainer.addSubview(clientFindButton);

// clientTextField.setOwner(this);
// panelContainer.addSubview(clientTextField);
clientListView = new ExtendedListView();
clientListView.setAllowsEmptySelection(true);
clientListView.setAllowsMultipleSelection(true);
// clientPopup = new Popup();

// clientPopup.addItem(PICK_ONE, null);
// panelContainer.addSubview(clientPopup);
clientScroll = new ScrollGroup();
clientScroll.setBuffered(true);
clientScroll.setHorizScrollBarDisplay(ScrollGroup.NEVER_DISPLAY);
clientScroll.setVertScrollBarDisplay(ScrollGroup.ALWAYS_DISPLAY);
clientScroll.setContentView(clientListView);
// clientScroll.setBackgroundColor(Color.lightGray);
panelContainer.addSubview(clientScroll);

// create the sector stuff
sectorLabel = new Label();
sectorLabel.setTitle(SECTOR);
sectorLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(sectorLabel);
sectorPopup = new Popup();
// sectorPopup.addItem(PICK_ONE, null);
panelContainer.addSubview(sectorPopup);
// create the priority stuff
priorityLabel = new Label();
priorityLabel.setTitle(PRIORITY);
priorityLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(priorityLabel);
priorityPopup = new Popup();
// priorityPopup.addItem(PICK_ONE, null);
panelContainer.addSubview(priorityPopup);

// create security popup list
securityLabel = new Label();
securityLabel.setTitle(SEcurity);
securityLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(securityLabel);
securityPopup = new Popup();
// securityPopup.addItem(PICK_ONE, null);
panelContainer.addSubview(securityPopup);

// create the comment stuff
commentLabel = new Label();
commentLabel.setTitle(COMMENT);
commentLabel.setJustification(Graphics.LEFT_JUSTIFIED);
panelContainer.addSubview(commentLabel);
// commentTextField = new CutCopyPasteTextField();
commentTextField = new CutCopyPasteTextView();
// commentTextField.setOwner(this);
commentTextField.setPopupMenuProvider(this);
commentScroll = new ScrollGroup();
commentScroll.setBuffered(true);
commentScroll.setHorizScrollBarDisplay(ScrollGroup.NEVER_DISPLAY);
commentScroll.setVertScrollBarDisplay(ScrollGroup.ALWAYS_DISPLAY);
commentScroll.setBackgroundColor(Color.white);
commentScroll.setContentView(commentTextField);

```

```
package service.database;
```

```
import java.lang.Thread;
import java.io.*;
import java.util.zip.*;
import java.util.*;
import java.sql.*;
import java.text.*;
import java.net.*;
```

```
import netscape.application.*;
```

```
import core.Interchange;
import COM.swissbank.notification_center.*;
import COM.swissbank.adapter.*;
import COM.swissbank.util.*;
```

```
import util.*;
import core.*;
import adapter.service.*;
import listener.*;
import listener.service.*;
import preferences.*;
import service.util.*;
```

```
public class BackChatService extends DatabaseService implements DatabaseListener {

    public static final boolean DEBUG = false;
    private static final String stSelectSQL = "spGetChannelLastMessages ?, ?";

    public BackChatService(){
        super();
    }

    protected String getDatabaseName(){
        return this.stDatabaseName;
    }

    protected String getDatabaseLogin(){
        return this.stDatabaseLogin;
    }

    protected String getDatabasePassword(){
        return this.stDatabasePassword;
    }

    protected String getDatabaseMachineName(){
        return this.stDatabaseMachineName;
    }

    protected String getDatabasePort(){
        return this.stDatabasePort;
    }

    protected void handleFileRecovery(Object _object){}

    protected void handleResultSet(ResultSet _resultSet) throws SQLException, IOException {
        try {
            if (DEBUG)
                System.out.println("BackChatService handleResultSet()");

            ContentAdapter _adapter =
                (ContentAdapter)Recycler.getSingleton().getObject(ContentAdapter.class);
        }
    }
}
```

```

BackChatService.java
while (_resultSet.next ()) {
    Calendar calendar = new GregorianCalendar(new
SimpleTimeZone(0, "GMT"));
    calendar.set(_resultSet.getInt(3),
                                                _resultSet.getInt(4) - 1,
                                                _resultSet.getInt(5),
                                                _resultSet.getInt(6),
                                                _resultSet.getInt(7));

    _adapter.addContent(new Content(_resultSet.getString(10).trim(),
        "#" + _resultSet.getString(11).toLowerCase().trim(),
        _resultSet.getString(9),
        calendar,
        Content.INTERCHANGE));
}
qResponses.push(_adapter);

if (DEBUG)
    System.out.println("BackChatService handleResultSet() finished");
}
catch(Exception _exception){
    _exception.printStackTrace();
}
}

protected void handleRequest(Object _object){
    DBObject _dbobject = (DBObject)_object;

    if(!DBObject.IRC.equals(_dbobject.getChannelType())) {
        Debug.report(this.getClass(), Debug.ASSERTION_DEBUG, "attempting to get backchat for a
non irc channel " + _dbobject.getChannel());
        return;
    }

    // BUG229
    // Modified by raduload on 6/23/99 - I will later replace the
    // static server location and cgi location (once its set in stone).
    if(Interchange.getInterchangeDataProvider().isRestrictedVersion()) {
        InputStream                                urlIn = null;
        URL                                         url = null;
        HttpURLConnection                          httpCon = null;
        ContentAdapter                            adapter = null;

        try {
            //core.Interchange.InterchangeDataProvider idp =
            core.Interchange.getInterchangeDataProvider();
            url = new URL("http://" + "hubirc" /*+ ":" + idp.getChatwebPort()*/ + "/" +
                Globals.BACKCHAT_ADDRESS + "?userid=" +
            Interchange.getInterchangeDataProvider().getUserNick()
                + "&count=30&chanid=" + _dbobject.getChannel().substring(1)); // +
            "&from=DATESTRING HTTP/1.0\n\n");
            Debug.report(this.getClass(), Debug.INFO_DEBUG, "http://" + "hubirc" /*+ ":" +
            idp.getChatwebPort()*/ + "/" +
                Globals.BACKCHAT_ADDRESS + "?userid=" +
            Interchange.getInterchangeDataProvider().getUserNick()
                + "&count=30&chanid=" + _dbobject.getChannel().substring(1));

            httpCon = (HttpURLConnection)url.openConnection();
            httpCon.connect();
            urlIn = httpCon.getInputStream();
            adapter = processInput(urlIn);
            if(adapter != null && adapter.getContent() != null) {
                qResponses.push(adapter);
            } else {
                Debug.report(this.getClass(), Debug.WARNING_DEBUG, "received no content for a
backchat request");
            }
            httpCon.disconnect();
            urlIn.close();

```

```

BackChatService.java
    } catch (Exception e) {
        Debug.report(this.getClass(), Debug.ERROR_DEBUG, "Error getting backchat for " +
        _dbobject.getChannel(), e);
    }
    } else {
        try{
            Debug.report(this.getClass(), Debug.INFO_DEBUG, "BackChatService handleRequest()");
            this.prepareJdbcConnection();

            if(this.connection == null) {
                try {
                    this.prepareJdbcConnection();
                } catch (Exception e) {
                    Debug.report(this.getClass(), Debug.ERROR_DEBUG, "database down");
                    e.printStackTrace();
                    return;
                }
            }
            if(this.connection == null) {
                Debug.report(this.getClass(), Debug.ERROR_DEBUG, "database connection down");
                return;
            }
        }

        this.preparedStatement = this.connection.prepareStatement(stSelectSQL);
        this.preparedStatement.setString(1, _dbobject.getChannel().substring(1));
        this.preparedStatement.setInt(2, 30);
        processStatement();
        Debug.report(this.getClass(), Debug.INFO_DEBUG, "BackChatService handleRequest()
finished");
    }
    catch(Exception _exception){
        Debug.report(this.getClass(), Debug.ERROR_DEBUG, _exception);
    }
}

/**
 * BUG229
 * Modified by raduload on 6/24/99
 *
 * Gets all the backchat date from a cgi.
 *
 * The format for the cgi's return of data needs to be changed because it
 * can cause problems now.
 *
 * @param input stream of backchat messages
 */
protected ContentAdapter processInput(InputStream input) {
    BufferedReader messagesReader = null;
    String inputLine = null;
    StringTokenizer tokenizer = null;
    ContentAdapter adapter = null;
    Calendar calendar = null;
    String author = null;
    String channel = null;
    String message = null;
    int year = 0;
    int month = 0;
    int day = 0;
    int hour = 0;
    int minute = 0;

    messagesReader = new BufferedReader(new InputStreamReader(input));

    try {
        adapter = (ContentAdapter)Recycler.getSingleton().getObject(ContentAdapter.class);

        while((inputLine = messagesReader.readLine()) != null) {
            Debug.report(this.getClass(), Debug.INFO_DEBUG, "BackChatService received: " +
inputLine);

```

```

BackChatService.java
if("").equals(inputLine)) {
    continue;
}
try {
    tokenizer = new StringTokenizer(inputLine, "~!~");

    // get the date of the message out
    calendar = new GregorianCalendar(new SimpleTimeZone(0, "GMT"));
    tokenizer.nextElement();
    tokenizer.nextElement();
    year = (new Integer(tokenizer.nextToken())).intValue();
    month = (new Integer(tokenizer.nextToken())).intValue() - 1;
    day = (new Integer(tokenizer.nextToken())).intValue();
    hour = (new Integer(tokenizer.nextToken())).intValue();
    minute = (new Integer(tokenizer.nextToken())).intValue();
    tokenizer.nextElement(); // seconds we ignore
    calendar.set(year, month, day, hour, minute);

    // get the actual message, channel and author out.

    author = tokenizer.nextToken().trim();
    channel = "#" + tokenizer.nextToken().toLowerCase().trim();

    message = tokenizer.nextToken("\0").substring(3);
    adapter.addContent(new Content(author, channel, message, calendar,
                                   Content.INTERCHANGE));
} catch (Exception e) {
    Debug.report(this.getClass(), Debug.ERROR_DEBUG, "error parsing a backchat
message" e);
}
} catch (Exception e) {
    Debug.report(this.getClass(), Debug.ERROR_DEBUG, e);
}
return(adapter);
}

protected void handleResponse(Object _object){
    try{
        Debug.report(this.getClass(), Debug.INFO_DEBUG, "handleResponse()");
        this.adapter = (ContentAdapter)_object;
        this.tnc.enqueueNotification((new
Notification(((ContentAdapter)(adapter)).getContent().getChannel(),(Object)adapter)));
        Debug.report(this.getClass(), Debug.INFO_DEBUG, "handleResponse() finished");
    }
    catch (Exception _exception){
        Debug.report(this.getClass(), Debug.ERROR_DEBUG, _exception);
    }
}

public void databaseTransaction(DBObject _dbObject) {
if(_dbObject == null) {
    if(DEBUG) {
        System.out.println("BackChatService databaseTransaction: _dbObject == null");
    }
    return;
}
/*
 * Only process backchat requests when they are of type read.
 * We don't write to backchat ever!
 */
if(_dbObject.getRequestType() == DBObject.READ){
if (_dbObject.getBackChatReload()) {
    this.qRequests.push(_dbObject);
    this.unlockService();
}
}
}

```

BackChatService.java

```
    }  
  
    public boolean performNotificationCommand(Notification n){  
        try{  
            ((Adapter)n.notificationObject()).apply(this);  
        }  
        catch(AdapterException _adapterException){  
            System.err.println("BackChatService(performNotification): " + _adapterException);  
            _adapterException.printStackTrace();  
        }  
        return true;  
    }  
}
```